Universidad de las Ciencias Informáticas
Facultad 4

Componente para acelerar el diseño de resortes helicoidales

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

Autor:
Gustavo García González

Tutores:
Dr.C Augusto César Rodríguez Medina
Ing. Sandy García Santos

La Habana, junio del 2016
"Año 58 de la Revolución"

Declaración de autoría

Declaro ser único autor del presente trabajo de diploma y autorizo a la Universidad de las Ciencias Informáticas a ser uso del mismo en su beneficio.
Para que así conste firmo la presente a los ____ días del mes de _____ del año 2016.

_____

Gustavo García González
Firma del Autor

_____ _____

Dr.C Augusto Cesar Rodríguez Medina
Firma del Tutor

Ing. Sandy García Santos
Firma del Tutor

"La mayoría de la gente se avergüenza de la ropa raída y de los muebles destartalados, pero más debería ruborizarse de las ideas nocivas y de las filosofías gastadas."

Albert Einstein

Dedicatoria:

A mis abuelos Jesús y Sara Rondina por ser los mejores abuelos del mundo, acompañarme espiritualmente y velar por mí en el camino para alcanzar mis sueños.

Agradecimientos:

A mis padres por todo el sacrificio que realizaron para que pudiera completar mis sueños y por apoyarme en los momentos más difíciles de mi vida.

A mi hermano Javier por ser insoportable pero quererme y ayudarme tal y como soy.

A Maydalis por ser la persona que colmó mi corazón de afecto, me soportó en los momentos más difíciles y me apoyó incondicionalmente en la mayoría de mi carrera.

A Magali, Fulgencio, Mayra y Sergio por acogerme en el seno de su familia, brindarme amor y apoyo cuando más lo he necesitado.

A Jessica, Yordy, Darián y todos los amigos que han compartido momentos anteriores en mi vida pero sin los cuales no hubiera llegado a donde estoy.

A Yisenia, Jorge, Julio y Ricardo por ser los mejores amigos que he tenido durante mi carrera y estar a mi lado cuando más he necesitado una mano amiga.

A todos mis amigos del antiguo y nuevo grupo por acompañarme en esta travesía y formar parte del periodo más hermoso de mi vida.

A todos los profesores que aportaron su gramo de arena a mi formación.

Resumen

El desarrollo de la informática ha proporcionado el surgimiento de una serie de sistemas para el Diseño Asistido por Computadora (CAD, por sus siglas en inglés). Algunos sistemas propietarios de esta rama poseen componentes que aceleran el diseño de entidades mecánicas complejas, como es el caso de los resortes helicoidales, los cuales son imposibles de adquirir por el bloqueo existente contra Cuba y los altos precios en el mercado internacional. Las aplicaciones de los resortes helicoidales van desde un simple interruptor eléctrico hasta la suspensión de un transbordador espacial o el mecanismo antivibratorio del telescopio de Monte Palomar, son innumerables sus usos industriales. El presente trabajo tiene como objetivo desarrollar un componente, con las herramientas disponibles de código abierto, para automatizar el diseño de resortes helicoidales que funcionen a compresión, tracción y torsión. La solución propuesta fue desarrollada con Open Cascade, Qt y SQLite. Para guiar el proceso de desarrollo se empleó como metodología el Proceso Ágil Unificado en su versión UCI. Con esta solución se reduce el costo de desarrollo de los resortes helicoidales en cuanto al proceso de diseño, se eliminan los problemas legales por posibles usos de sistemas propietarios de forma indebida y se da un paso más en la soberanía tecnológica del país.

Palabras claves: diseño, industria, ingeniería, resortes.

Índice

Introducción

El presente trabajo posee sus bases en un proyecto de la Facultad 4 de la Universidad de las Ciencias Informáticas, el cual contiene como propósito desarrollar una aplicación CAD, destinada a los sectores de la industria nacional vinculados a la actividad de diseño. En este contexto una de las tareas fue automatizar el proceso de modelado geométrico y cálculos de las propiedades físicas de los resortes helicoidales de compresión, torsión y tracción utilizando la tecnología *Open Cascade* y el *framework* Qt.

Este documento está estructurado en tres capítulos:

- Capítulo 1: Se aborda la fundamentación teórica de la investigación, donde se incluye un estudio de los sistemas CAD homólogos y sus módulos, las propiedades y características de los resortes helicoidales en cuestión, las metodologías y herramientas para el desarrollo.
- Capítulo 2: Se muestra el modelo del dominio, la descripción del componente, la arquitectura, patrones de diseño, requisitos, entre otros elementos.
- Capítulo 3: Se presenta los estándares de codificación, diagrama de componente y las pruebas realizadas.

1. Fundamentación teórica

En este capítulo se exponen aspectos iniciales sobre la situación problémica y el diseño de la investigación a realizar. Además, recoge información esencial de sistemas homólogos y de los resortes helicoidales de compresión, torsión y tracción. También, aborda temas como las metodologías y las herramientas de desarrollo a emplear.

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

El diseño mecánico es el esbozo de objetos y sistemas de naturaleza mecánica, piezas, mecanismos, máquinas, instrumentos y dispositivos diversos. Hace uso de las matemáticas, la ciencia de los materiales y las ciencias mecánicas aplicadas a la ingeniería.(1)

Entre los siglos XIX y XX las tecnologías para el dibujo estaban confinadas a una estructura básica, compuesta por el proyectista, con el tablero de dibujo y sus instrumentos de trazado. Precisamente estos tableros han sufrido transformaciones para tratar de facilitar la productividad y el desempeño de los proyectistas.(2)

Estas tecnologías para el dibujo, dieron un salto significativo con la aparición de los sistemas de cómputo digital, surgiendo así un nuevo término Diseño Asistido por Computadora.

“El primer CAD data de los años 50 por las Fuerzas Aéreas de los Estados Unidos de América. El primer sistema de gráficos, el SAGE (*Semi Automatic Ground Environment*, en español Entorno Terrestre Semiautomático) un sistema de defensa aérea, que fue empleado para visualizar datos de radar, fue desarrollado en colaboración con el Instituto Tecnológico de Massachusetts. En los 60, los sistemas CAD se utilizaron para diseñar espacios interiores de oficinas. En 1968 estaban ya disponibles los sistemas CAD 2D (muy básico, tal y como lo entendemos hoy día). Estos sistemas funcionaban en terminales de grandes ordenadores (*mainframes*).” (3)

“A principios de los 70 varias compañías empezaron a ofrecer sistemas de diseño/dibujo automatizado. Muchos de los productos y firmas más conocidas en la actualidad tuvieron sus inicios en este período. Podían encontrarse ya algunas capacidades 3D en programas de cálculo de sistemas HVAC (*Heating, Ventilation and Air Conditioning,* en español Calefacción, Ventilación y Aire Acondicionado). A finales de los 70, un sistema típico CAD consistía en un mini-ordenador de 16 bits con un máximo de 512 Kb de memoria y de 20 a 300 Mb de disco duro”. (3)

En los 80 la empresa Autodesk entra en escena con el objetivo de crear un programa CAD que funcionara sobre una PC. En poco tiempo Autodesk AutoCAD llegó a ser el programa más popular. Muchos otros programas de compañías diversas siguieron la misma senda. Durante esta década, los programas CAD se utilizaban básicamente para desarrollos de ingeniería. Además, en esta etapa empiezan a desarrollarse los sistemas GIS (*Geographical Information Systems*, en español Sistema de Información Geográfica). (3)

A mediados de los 90 aparecen muchos programas CAD para una gran variedad de usos y aplicaciones. A finales de los 90 mucha gente utiliza ya los programas CAD de forma habitual, pero hay todavía una gran lucha por atraer la atención de los usuarios. En este período se desarrollan programas mejores para satisfacer las necesidades crecientes de la industria y se ofrecen soluciones para la construcción, ingeniería civil, mecánica, entre otros sectores.(3)

Estos programas permiten que el usuario pueda asociar a cada entidad una serie de propiedades como color, capa, estilo de línea, nombre, definición geométrica, material, entre otros, que permiten manejar la información de forma lógica. Además, se pueden renderizar los modelos 3D para obtener una previsualización realista del producto, aunque a menudo se prefiere exportar los modelos a programas especializados en visualización y animación.

En la actualidad hay una amplia gama de programas CAD propietarios, como AutoDesk Incorporated, Dassault Systèmes, Solid Edge, entre otros. Debido al bloqueo impuesto sobre Cuba por parte de los Estados Unidos de América se hace imposible la compra y uso de estas tecnologías en la industria cubana, por lo que es imposible la comercialización de proyectos desarrollados con estas herramientas de forma ilegal.

Entre las herramientas CAD de código abierto que podría optar un ingeniero para su uso, con el fin de eliminar las dificultades antes mencionadas, están el LibreCAD y el FreeCAD, pero estas no cuentan con los aceleradores de diseño[1] que poseen sus homólogos propietarios, por lo que el modelado de las estructuras complejas se realizan con un mayor grado de dificultad y con un elevado tiempo de trabajo.

Un componente común en algunos de los sistemas CAD propietarios es el encargado de la representación de los resortes helicoidales de compresión, tracción y torsión.

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

“Un resorte mecánico, es un elemento de máquina que posee la capacidad de acumular energía mecánica para liberarla oportunamente con el fin de ejercer fuerza, brindar flexibilidad o reducir vibraciones. La variedad de sus formas es muy amplia y sus aplicaciones que van desde un simple interruptor eléctrico hasta la suspensión de un transbordador espacial o el mecanismo antivibratorio del telescopio de Monte Palomar, son innumerables”. (5)

Los resortes de compresión poseen muchos tipos de aplicaciones como empujar o torcer, así le permite lograr numerosos resultados. Ofrecen resistencia a fuerzas lineales de compresión (empujón) y son en realidad uno de los más eficientes dispositivos de almacenamiento de energía. Una pluma es un excelente ejemplo de cómo los resortes de compresión pequeños trabajan. El pequeño resorte comprimirá cuando la pluma es hecha clic y entonces el pequeño resorte regresará a su posición original. Otras de las aplicaciones más comunes son en los amortiguadores de los autos, trenes y otros vehículos.

Otro de los resortes helicoidales son los de tracción, los cuales son empleados a gran escala en la industria automotriz en los mecanismos de freno, limpiaparabrisas, en las ventanas, en las puertas de los pasajeros y de los compartimientos de motores, etc.

Los resortes de tracción se encuentran en:

1. Sistemas de audio como Cd o casete
2. Puertas para hornos y cocinas integrales
3. Tostadoras
4. Pinzas para ropa
5. Interruptores de corto circuito

Los resortes helicoidales que trabajan a torsión son empleados en las pinzas de ropa, las trampas para ratones, mecanismos de contrapeso tradicionales, en las puertas de los garajes y en los sistemas de apertura de maleteros de coches. Algunos pequeños resortes de este tipo se usan para equipos electrónicos, por ejemplo en las tapas de las cámaras digitales.

Teniendo en cuenta la imposibilidad de adquirir sistemas de alta tecnología como las herramientas CAD propietarias, la insuficiencia de los sistemas de software libre (FreeCAD y LibreCAD) al no poseer un componente que acelere el modelado de los resortes helicoidales, la necesidad de una aplicación que garantice la soberanía tecnológica y los disímiles usos en la industria de estos elementos mecánicos complejos, se define como **problema a resolver:**

Inexistencia de un componente para automatizar el diseño de resortes helicoidales en el contexto de la ingeniería y el diseño a nivel nacional, como parte de una plataforma informática legalmente adquirida.

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

Para la solución del problema antes mencionado se propone como **objetivo general:**

Desarrollar un componente, con las herramientas disponibles de código abierto, para automatizar el diseño de resortes helicoidales que funcionen a compresión, tracción y torsión.

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

1. Diseñar el componente.
2. Obtener un componente funcional capaz de automatizar el proceso de diseño de los distintos tipos de los resortes helicoidales.

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

1. Asimilación de los principales conceptos y tecnologías que se requieren para el desarrollo del componente (lenguaje C++, *framework* de Qt y la tecnología Open Cascade).
2. Definición del perfil y diseño de la investigación.
3. Fundamentación teórico-matemática de los resortes helicoidales.
4. Estudio de las investigaciones previas.
5. Análisis de los módulos disponibles en Autodesk Inventor y Solid Edge.
6. Consultar las metodologías de diseño de resortes helicoidales con el propósito de establecer las variables y parámetros necesarios para el diseño de los mismos.
7. Selección de la metodología de desarrollo a emplear.
8. Selección del sistema gestor de base de datos a emplear para almacenar los resortes helicoidales y los materiales normalizados.
9. Obtención de los requisitos a partir de los sistemas propietarios y las metodologías de diseño de los resortes helicoidales.
10. Definición de la arquitectura del componente.
11. Definición de los patrones de diseño con los que cumplirá el componente.
12. Elaboración de la estructura de clases que conformarán el componente.

13. Modelado del flujo de datos del componente.
14. Diseño de la interfaz gráfica del componente.
15. Implementación del componente destinado a modelar resortes helicoidales de compresión, tracción y torsión.
16. Comprobación del componente implementado.

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

Métodos teóricos:

- **Análisis y síntesis:** se empleó para la construcción y desarrollo de la teoría, para la profundización en el tema y la sistematización del conocimiento.
- **Modelación:** se hizo uso de este método para la representación abstracta de los elementos que se investigan, como por ejemplo en el modelado del dominio realizado.

Métodos empíricos:

- **Observación:** se empleó para caracterizar las soluciones, teniendo en cuenta distintos datos y tipos de resorte, de otras aplicaciones CAD (FreeCAD, SolidEdge, AutoCAD Inventor), y así establecer los requisitos con las principales funcionalidades de estos.
- **Experimentación:** para comprobar que el modelado sea el adecuado, a partir de la forma del objeto y las dimensiones del mismo.
- **Medición:** durante los experimentos o pruebas de funcionamiento se toman los tiempos de modelado y cálculo para poder estimar de manera objetiva la eficiencia del módulo.

1.2 Resortes helicoidales

“En el diseño de la mayoría de los elementos mecánicos es deseable, que la deformación inducida por el estado de cargas actuante sea lo más baja posible. Sin embargo, los resortes mecánicos cumplen en las máquinas la misión de elementos flexibles, pudiendo sufrir grandes deformaciones por efecto de cargas externas volviendo a recuperar su forma inicial cuando cesa la acción de las mismas, es decir, presentan una gran elasticidad.”(6)

“Para su fabricación se emplean aceros de gran elasticidad (acero al carbono, acero al cromo-vanadio, acero al cromo-silicio, etc.), aunque para algunas aplicaciones especiales pueden utilizarse el cobre endurecido y el latón.”(6)

“Debido al elevado valor de la maquinaria en la industria, los resortes han sido estudiados con meticulosidad; además, se producen en masa y se han determinado configuraciones ingeniosas para lograr una variedad de propiedades deseadas.”(6)

Las aplicaciones de los resortes son muy variadas entre las más importantes pueden mencionarse las siguientes(6):

- Como elementos capaces de absorber energía o cargas de choque, como por ejemplo en chasis y topes de ferrocarril.
- Como dispositivos de fuerza para mantener el contacto entre elementos, tal como aparece en los mecanismos de leva y en algunos tipos de embragues.
- En sistemas de suspensión y/o amortiguación, percibiendo la energía instantánea de una acción externa y devolviéndola en forma de energía de oscilaciones elásticas.
- Como elemento motriz o fuente de energía, como en mecanismos de reloj y juguetes, dispositivos de armas deportivas, etc.
- Como elementos capaces de absorber vibraciones.

1.2.1 Fundamentación matemática de la curva base para diseñar resortes

Los resortes helicoidales están definidos por una curva llamada hélice, la cual es “una curva donde la tangente hace un ángulo constante con una línea fija”. Estas pueden ser en dirección derecha o izquierda. (7)

La hélice es una curva en el espacio que posee como ecuación paramétrica(7):

$$x = r * \cos(t)$$
$$y = r * sen(t) \qquad (1\text{-}1)$$
$$z = c * t$$

donde $t \in [0; 2\pi]$, y r es el radio de la hélice y c es la constante que da la separación vertical de las vueltas de la misma.

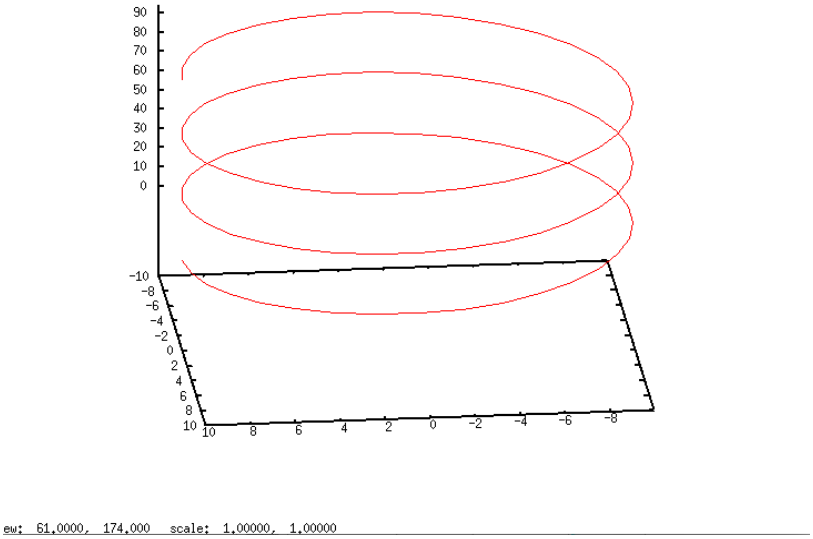


Fig. 1 Hélice realizada usando la ecuación paramétrica en wxMaxima con un radio de 10 u, paso de 5 u y 3 espiras.

La curvatura está dada por la ecuación(7):

$$k = \frac{r}{r^2+c^2} \qquad (1\text{-}2)$$

La longitud del arco está dada por(7):

$$s = \sqrt{r^2 + c^2} * t \qquad (1\text{-}3)$$

La ecuación de la torsión es(7):

$$\tau = \frac{c}{r^2+c^2} \qquad (1\text{-}4)$$

1.2.1.1 Resortes helicoidales de compresión

"Los resortes helicoidales sometidos a cargas de compresión, se caracterizan por tener un ángulo de inclinación **α** diseñado de modo que durante la operación sus espiras no entren en contacto y porque a diferencia de los resortes de tracción, eventualmente pueden fallar por pandeo y no necesariamente deben contar con un elemento adicional que permita transmitir la carga desde el soporte hasta el cuerpo del resorte."(5)

En la figura A.1 del Anexo A, se muestran distintos estados de un resorte de compresión. Cuando el mismo está descargado, L_0 indica la longitud libre. Al ser sometido a una carga **P**, el resorte se deforma una longitud **f** y con L_c se indica la longitud comprimida: $L_c = L_0 - f$.

“Cuando la deformación es máxima, de modo que cada espira del resorte está en contacto con la siguiente, entonces la longitud comprimida se denomina Longitud Sólida y se indica con **Ls**.”(5)

En la figura A.2 del Anexo A, se muestran los cuatro tipos de extremos que habitualmente se utilizan en resortes de compresión.

“Los extremos simples, se obtienen sencillamente al cortar el resorte en dos puntos” (4) (Fig. A.2.a). Este método presenta la ventaja de su bajo costo de producción pero tiene la desventaja que solo apoya un punto del extremo en el soporte con un ángulo de inclinación **α,** por lo que la transferencia de carga no es óptima. Para solucionar esta desventaja funcional puede doblarse la espira libre hasta un ángulo de inclinación de cero grados. A este tipo de extremo se lo llama escuadrado o cerrado”(5) (Fig. A.2.b)

Otra forma de mejorar la transferencia de carga consiste en aplanar por maquinado (con una amoladora por ejemplo) la espira libre, obteniendo un extremo simple amolado”(5) (Fig. A.2.c)

Al tipo de extremo que combina las dos soluciones anteriores (Fig. A.2.d) se lo denomina escuadrado amolado.

1.2.1.2 Resortes helicoidales de tracción

“Estos resortes, sometidos a cargas de tracción, se caracterizan por tener un ángulo de inclinación **α** muy pequeño, de modo que al estar descargados sus espiras suelen estar en contacto, y porque deben contar con un elemento que permita transmitir la carga desde el soporte hasta el cuerpo del resorte.” (5)

“Este problema inicial de diseño puede resolverse colocando un dispositivo externo en los extremos del resorte como puede ser un tapón roscado o un gancho giratorio, pero de esta forma en el proceso de fabricación se incrementaría considerablemente el costo del producto terminado, por lo que habitualmente se fabrica un gancho fijo en los extremos del resorte con el mismo alambre de las espiras extremas según alguno de las formas indicadas” (4) en la Figura A.3 del Anexo A. “En caso de diseñar un resorte con extremo de gancho fijo, debe considerarse el efecto de concentración de tensiones en el doblez del mismo.”(5)

1.2.1.3 Resortes helicoidales de torsión

“Estos resortes se someten a la acción de un momento flector:

$$M = F * r \quad (1\text{-}5)$$

que produce una tensión normal en el alambre. Las tensiones remanentes debidas al enrollado hacen más fuerte al resorte; se diseñan para que funcionen a niveles de tensión que sean iguales o excedan la resistencia de fluencia del material."(8) Para mejorar el uso de estos se han desarrollado distintos tipos de extremos, con el fin de explotar las características tensionales de los mismos (Fig. A.4 del Anexo A)

1.2.2 Fatiga

Los resortes helicoidales están casi siempre sujetos a cargas de fatiga, la cual es el fallo de un componente como resultado de la tensión cíclica. La fatiga depende de muchos factores, incluyendo características fundamentales de las materias primas, la magnitud y la orientación de los esfuerzos aplicados, la historia de procesamiento, etc. Las fallas por fatiga a menudo son el resultado de la tensión aplicada a niveles significativamente inferiores a las necesarias para causar un fallo estático.(9)

"En muchos casos el número de ciclos de vida requerido para fatigar un resorte puede ser pequeño, por ejemplo, varios miles para un resorte de candado. Sin embargo, el resorte de la válvula de un motor de automóvil debe sostener millones de ciclos de operación sin fallo, por lo que debe estar diseñado para la vida infinita."(10)

La existencia de un límite de resistencia a la fatiga (o límite de fatiga) para los aceros se descubre por el científico alemán August Wohler, quien realizó pruebas sobre probetas de acero sometidas a "flexión giratoria", denominada así al tipo de carga que se genera en un elemento que gira sometido a un momento flector constante. En dichas pruebas se pretendía relacionar los niveles de esfuerzo a los cuales se sometían las probetas, con el número de ciclos de carga que soportaban hasta la falla. Wohler obtuvo un diagrama como el de la figura 2, el cual es conocido como diagrama S-n_c (esfuerzo - número de ciclos) o diagrama de vida-resistencia de Wohler. El esfuerzo (o resistencia) S corresponde al valor del esfuerzo máximo al cual se somete la probeta, y n_c es el número de ciclos de esfuerzo. Las líneas del diagrama representan aproximaciones a los puntos reales de falla obtenidos en los ensayos. (11)

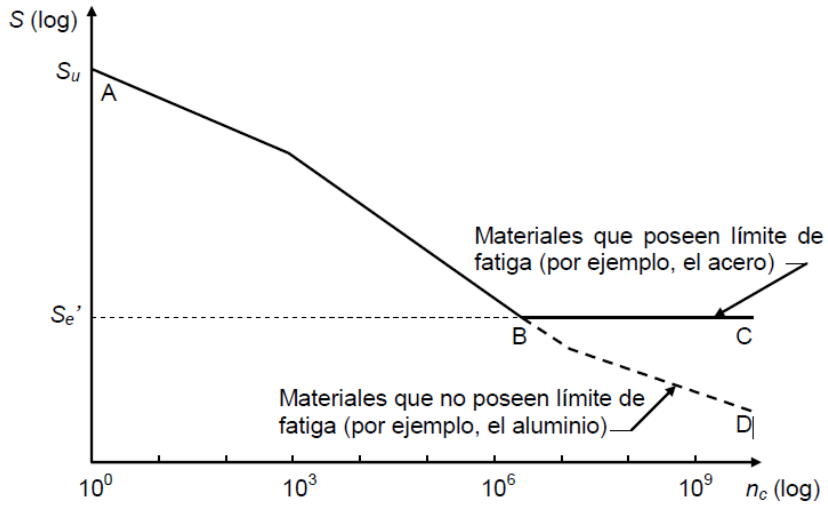


Fig. 2 Diagrama S-nc o diagrama de Wohler(11).

El diagrama para muchos aceros es como el dado por la curva ABC. La curva tiene un codo en *S* = Se' (en este caso S es el equivalente a Sf, resistencia a la fatiga[2]) y $nc \approx 10^6$ ciclos, a partir del cual el esfuerzo que produce la falla permanece constante. Esto indica que si la probeta se somete a un esfuerzo menor que *Se'*, ésta no fallará; es decir, la probeta tendrá una vida infinita. A niveles superiores de esfuerzo, la probeta fallará después de un número de ciclos de carga y, por lo tanto, tendrá vida finita. Como *Se'* es el límite por debajo del cual no se produce falla, se le conoce como límite de fatiga (esfuerzo máximo invertido que puede ser repetido un número indefinido de veces sobre una probeta normalizada y pulimentada sometida a flexión, sin que se produzca falla o rotura). (11)

Definiendo la resistencia a la fatiga en un número de ciclos como(10):

$$(S'f)n = \frac{E\Delta\varepsilon_e}{2} \qquad (1\text{-}6)$$

se puede escribir la ecuación $\frac{\Delta\varepsilon}{2} = \frac{\sigma'f}{E}(2N)^c$ como(10):

$$(S'f)n = \sigma'f(2N)^b \quad (1\text{-}7)$$

A 10^3 ciclos la ecuación anterior se puede escribir como(10):

$$(S'f)_{10^3} = \sigma'f(2 * 10^3)^b = fSut \qquad (1\text{-}8)$$

[2] La resistencia a la fatiga para vida finita es una propiedad que se basa en pruebas de flexión giratoria sobre probetas normalizadas y pulidas.(11)

donde f es la fracción de Sut representada por $(S'f)_{10^3}$ y Sut es la fuerza tensora máxima, quedando que(10):

$$f = \frac{\sigma' f}{Sut}(2 * 10^3)^b \qquad (1\text{-}9)$$

Sin embargo, al menos que los datos actuales estén disponibles, se recomienda el uso de los valores de la Fig. 3 para f. (10)

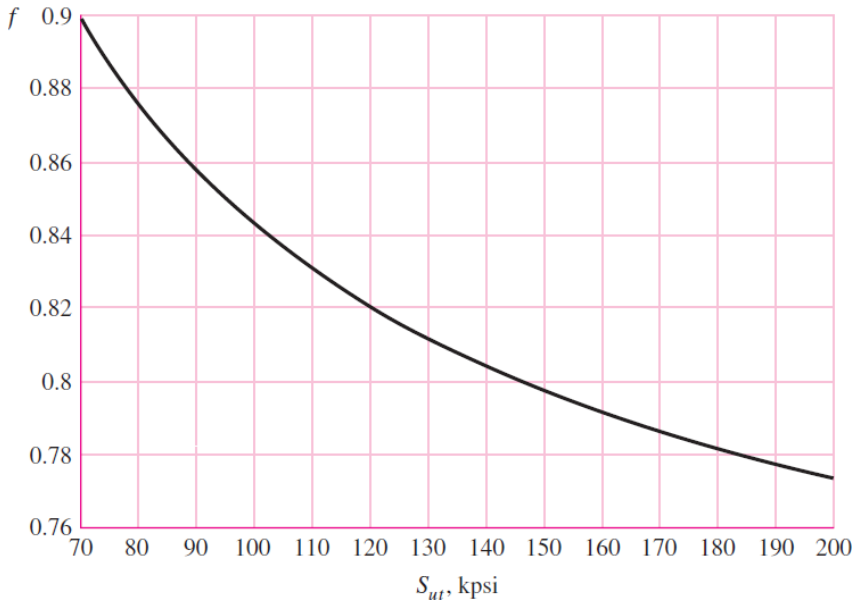


Fig. 3 Fracción de la fuerza a la fatiga (f)(10).

La ecuación de $(S'f)n$ para la fatiga a ciclos medios $(10^3 \leq N \leq 10^6)$ en componentes mecánicos actuales se puede escribir como(10):

$$Sf = aN^b \qquad (1\text{-}10)$$

donde,

$$a = \frac{(f\ Sut)^2}{Se} \qquad (1\text{-}11)$$

y

$$b = -\frac{1}{3}\log\left(\frac{fSut}{Se}\right) \qquad (1\text{-}12)$$

Como se pudo apreciar en la Fig. 2 la fatiga a ciclo bajo $(1 \leq N \leq 10^3)$ se comporta casi linealmente. En la línea recta entre 1 y 10^3 Sut es conservativo y puede ser dado como(10):

$$Sf \geq SutN^{(\log f)/3} \qquad (1\text{-}13)$$

1.2.2.1 Líneas de falla

El límite de fatiga y la resistencia a la fatiga para vida finita constituyen propiedades base para el diseño de elementos sometidos a cargas variables, como es el caso de los resortes helicoidales. Sin embargo, no puede ser utilizado directamente en el diseño, ya que es obtenido bajo condiciones especiales de esfuerzo: probeta normalizada y pulida, girando sometida a flexión bajo condiciones ambientales favorables.(11)

Bajo una carga de flexión, la probeta sufre una variación sinusoidal de esfuerzo repetido invertido para la cual el esfuerzo medio es igual a cero, por lo que se necesitan ecuaciones de diseño que sirvan no sólo para un esfuerzo repetido invertido, sino también para cualquier tipo de variación sinusoidal. Para encontrar dichas ecuaciones fueron necesarias más pruebas experimentales, de las cuales se concluyó que, en general, si se agrega una componente media del esfuerzo, el elemento falla con una componente alternativa menor. (11)

Las figuras 4 y 5 “muestran las tendencias típicas que siguen los puntos de ensayo en diagramas de esfuerzo medio - esfuerzo alternativo, S_m-S_a y S_{ms}-S_{as}, respectivamente, cuando se someten probetas normalizadas y pulidas a diferentes combinaciones de estos esfuerzos. Los cruces en los diagramas indican las combinaciones (S_m, S_a) o (S_{ms}, S_{as}) para las cuales un pequeño aumento en el esfuerzo medio o en el alternativo produciría la falla de la probeta. Las combinaciones de esfuerzos que estén entre la zona de las cruces y el origen del diagrama no producirían falla, mientras que combinaciones de esfuerzos que estén por fuera de la zona de las cruces producirían la falla de la probeta de ensayo.”(11)

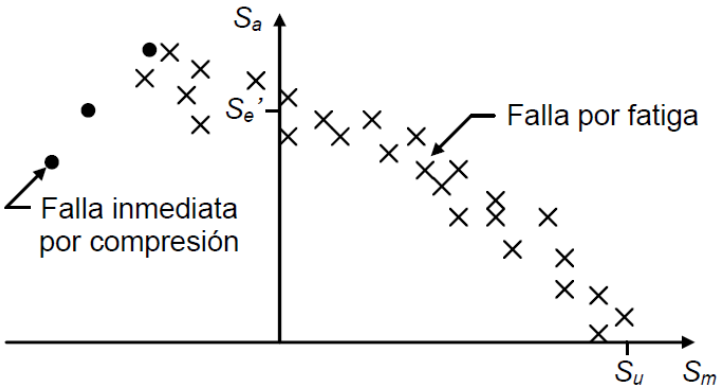


Fig. 4 Diagrama Sm-Sa (esfuerzos normales)(11)

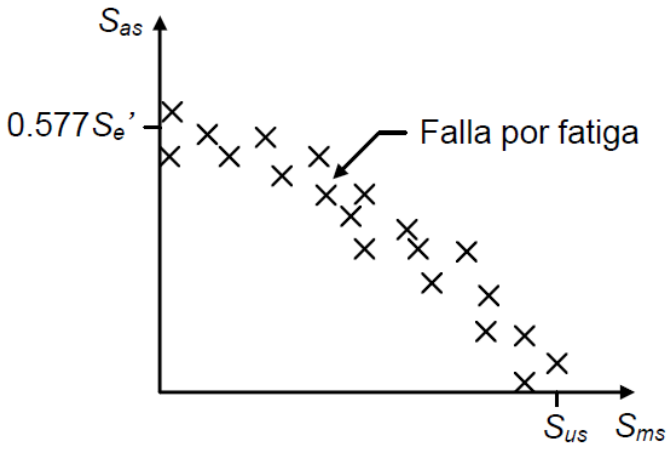


Fig. 5 Diagrama Sms-Sas (esfuerzos cortantes)(11)

En esta investigación Sas es equivalente a τ_a, Sms al τ_m, el Sus al Ssu, Se' al Se y Su al Sut.

Para modelar los datos experimentales expuestos en los diagramas anteriores, se han propuesto, entre otros, tres modelos o aproximaciones diferentes, las cuales se muestran en la figura 6:

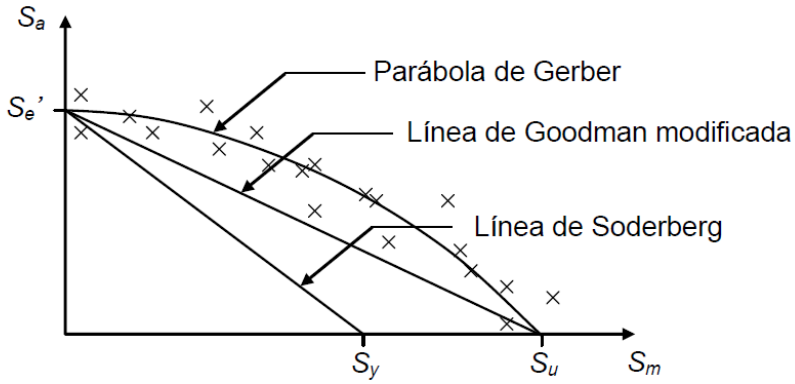


Fig. 6 Líneas (modelos) de falla en el diagrama esfuerzo medio contra esfuerzo alternativo(11)

En la presente investigación se prefiere emplear la línea de Goodman modificada por ser "la más utilizada en el diseño por fatiga"(10), la cual posee como función característica(11):

$$1 = \frac{Sm}{Sut} + \frac{Sa}{Se} \qquad (1\text{-}14)$$

donde Sm y Sa son el esfuerzo promedio y el esfuerzo alternativo respectivamente.

Esta aproximación se puede emplear con los esfuerzos cortantes escribiéndose como(10):

$$1 = \frac{\tau_a}{Sse} + \frac{\tau_m}{Ssu} \qquad (1\text{-}15)$$

siendo $Ssu = 0.67Sut$ y $Sse = 0.577Se$.

1.2.3 Estándares internacionales de resortes helicoidales

"Una norma es un documento que especifica los requisitos para los productos, servicios y/o procesos, donde se establecen las características requeridas. Esto ayuda a garantizar la libre circulación de mercancías y anima a las exportaciones. La normalización promueve la garantía de eficiencia y calidad en la industria, la tecnología, la ciencia y el sector público. Sirve para proteger a las personas y bienes, y para mejorar la calidad en todos los ámbitos de la vida. Las normas se elaboran en un proceso organizado basado en el consenso por un organismo de normalización reconocido."(12)

En la actualidad existen miles de catálogos de normas realizados por instituciones normalizadoras, como la Organización Internacional de Normalización (ISO, por sus siglas en inglés) y el Instituto Alemán de Normalización (DIN, por sus siglas en alemán), de componentes mecánicos, entre ellos los resortes, que definen las características de los mismos, así como el material del que están fabricados.(Fig. A.17 del Anexo A)

1.3 Módulos y componentes para el modelado y cálculo de resortes helicoidales.

En la actualidad con el crecimiento de las ciencias informáticas, el hardware de las nuevas computadoras y el uso de la Internet, han surgido un grupo de sistemas con el objetivo de satisfacer las necesidades de los ingenieros y diseñadores mecánicos en su trabajo cotidiano, facilitándoles el trabajo de diseño en múltiples planos de trabajoso entendimiento y cálculos de las propiedades físicas de los distintos objetos a modelar, reduciendo el tiempo de trabajo y aumentando la calidad del mismo. En este apartado se abordarán algunos de estos sistemas, algunos de escritorio y otros que se encuentran en la web.

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

1.3.1.1 Autodesk Inventor

"Autodesk Inventor permite a los fabricantes ir más allá de un simple modelo 3D, permitiéndoles a los ingenieros y/o diseñadores crear prototipos digitales, gracias a un

conjunto de herramientas para diseño mecánico 3D, simulación, análisis, visualización, y documentación. Con Inventor los ingenieros pueden integrar los planos 2D de AutoCAD y los modelos 3D en un único modelo digital, creando representaciones digitales del producto final que les permite validar la forma, dimensiones, precisión del ensamble (tolerancias), así como el comportamiento del producto antes que sea construido." (13)

Este programa propietario solo se puede usar en Windows y MacOS. Posee tres módulos independientes en los que se pueden modelar y mostrar los cálculos de las propiedades del resorte.

Para el modelado de resortes de compresión, en la versión del 2009, se escoge la cantidad de vueltas muertas en los extremos y se pueden seleccionar los datos a entrar inicialmente, siendo calculados los otros a partir de estos. Luego se muestra la sección de cálculo de otras propiedades, como la fuerza, los datos del material que se empleará en su construcción (el cual puede ser escogido de una base de datos), la masa, la fatiga, entre otros. (Fig. A.5 y A.6 del Anexo A)

De la misma manera, antes descrita funciona con el resorte de extensión y el de torsión, pero para el de extensión se escoge el tipo de extremo derecho e izquierdo, y para el de torsión el ángulo de inclinación de las piernas.

1.3.1.2 Solid Edge

"Solid Edge es un portafolio de herramientas de software asequible y fácil de usar que abordan todos los aspectos del proceso de desarrollo de productos - diseño en 3D, simulación, fabricación, gestión del diseño y mucho más gracias a un creciente ecosistema de aplicaciones. Solid Edge combina la velocidad y simplicidad del modelado directo con la flexibilidad y el control de diseño paramétrico - hecho posible con *synchronous technology*[3]." (15)

Este programa propietario solo se puede usar en Windows y MacOS. En su versión ST4, a diferencia de Autodesk Inventor, solo posee dos módulos independientes en los que se pueden modelar y mostrar los cálculos de las propiedades físicas del resorte (no posee el componente de modelado de resortes de torsión).Estos módulos brindan pocos cálculos, provocando la posibilidad que el usuario no encuentre toda la información necesaria.

En este se puede seleccionar el tipo de metodología a emplear, variando así las variables iniciales a introducir, siendo las otras calculables a partir de estas. Una vez introducidos los

[3] Synchronous Technology es la combinación de la velocidad y simplicidad del modelado directo con la flexibilidad y el control del diseño paramétrico.(14)

datos, incluyendo la selección del material, se calculan las propiedades del resorte señalando si es un resorte factible o no.

1.3.1.3 CATIA

"CATIA ofrece la posibilidad, no solo de modelar cualquier producto, sino de hacerlo en el contexto de su comportamiento en la vida real: diseño en la era de la experiencia. Los arquitectos de sistemas, los ingenieros, los diseñadores y todos sus colaboradores pueden definir el mundo que nos conecta, imaginarlo y darle forma." (16)

CATIA presenta las siguientes características:

- "Entorno de diseño social basado en una fuente única de autenticidad, al que se accede mediante potentes paneles en 3D que impulsan la inteligencia empresarial, el diseño simultáneo en tiempo real y la colaboración de todas las partes interesadas, incluidos los trabajadores móviles." (16)
- "3DEXPERIENCE ofrece una experiencia intuitiva con funcionalidades de modelado y simulación en 3D." (16)
- "Se trata de una plataforma inclusiva de desarrollo de productos, que resulta fácil de integrar con los procesos y las herramientas existentes. "(16)

Está disponible para Microsoft Windows, Solaris, IRIX y HP-UX.

La versión 5 de esta aplicación, a diferencia de los dos sistemas antes descritos, no ofrece un acelerador de diseño de los resortes helicoidales, por lo que para el modelado de los mismos hay que utilizar la primitiva de la hélice y en una ventana dar el paso de la misma, la dirección, el ángulo de inicio y otras características.

Para crear la forma del resorte hay que realizar el perfil del alambre y aplicando un "*Rib*"[4] se rota por la hélice.

En este caso, es bastante complejo la realización de los resortes de extensión y torsión, siendo esta la principal desventaja para su uso.

1.3.1.4 SolidWorks

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

[4] Rib es el nombre de la funcionalidad empleada en CATIA para realizar el barrido de un perfil.

El programa permite modelar piezas y conjuntos y extraer de ellos tanto planos técnicos como otras informaciones necesarias para la producción.

Este posee un grupo de productos con distintas funcionalidades, entre los que se encuentran:

- Soluciones CAD 3D: permite a las empresas acelerar el desarrollo de productos, reducir los costes de fabricación, y mejorar la calidad y fiabilidad del producto en una gran variedad de sectores y aplicaciones.(8)
- Visualización: proporciona un conjunto de herramientas de software independientes que combinan las funciones de renderizado líderes del sector con unas características y flujos de trabajo orientados al diseño, las cuales facilitan la creación fácil y rápida de contenido visual a diseñadores, ingenieros, expertos de *marketing* y otros creadores de contenidos.(17)
- Simulación: brinda una herramienta para realizar pruebas con una amplia variedad de parámetros (durabilidad, respuesta dinámica y estática, movimiento del ensamblaje, transferencia de calor, dinámica de fluidos y moldeo de plásticos por inyección) durante el proceso de diseño.(18)
- Gestión de datos de productos: permite tener control sobre los datos de diseño y mejorar considerablemente la manera en que sus equipos administran y colaboran en el desarrollo de productos.(19)
- Diseño eléctrico: proporciona una serie de funcionalidades de diseño de sistemas eléctricos para satisfacer las necesidades de los profesionales.(20)

En la versión del 2012 este sistema no posee, al igual que CATIA, de un acelerador gráfico para el modelado de resortes helicoidales, para poder realizar un resorte hay que realizar la primitiva de la hélice (Fig. A.10 del Anexo A). Posteriormente se busca el plano en el que se va a colocar el perfil del alambre del resorte (Fig. A.11 del Anexo A), se realiza el mismo y finalmente se aplica un barrido (Fig. A.12 del Anexo A).

Como se puede apreciar, al igual que en CATIA, es trabajoso realizar el diseño de los resortes helicoidales, sobre todo los de extensión y torsión que conllevan varias partes fusionadas.

1.3.1.5 FreeCAD

"FreeCAD es un modelador CAD 3D. El desarrollo es completamente de código abierto (licencia LGPL[5]). FreeCAD está dirigido directamente a la ingeniería mecánica y diseño de productos, pero también se emplea en una amplia gama de usos en la ingeniería, como la arquitectura."(22)

"FreeCAD cuenta con herramientas similares a CATIA, SolidWorks o Solid Edge, y por lo tanto también cae en la categoría de MCAD, PLM, CAx y CAE. Es un modelador paramétrico con una arquitectura de software modular que hace fácil proporcionar funcionalidades adicionales sin modificar el sistema central. Al igual que muchos modeladores modernos CAD 3D tiene componentes 2D con el fin de esbozar formas en dos dimensiones o extraer los detalles del diseño del modelo de 3D para crear dibujos de producción en 2D."(22)

FreeCAD hace un uso intensivo de todas las grandes bibliotecas de código abierto que existen en el campo de la Computación Científica. Entre ellas se encuentran Open Cascade, un núcleo CAD de gran alcance, Coin3D, una encarnación de Inventor abierto, Qt y Python. FreeCAD también es totalmente multiplataforma, y actualmente se ejecuta sin problemas en los sistemas Windows, Linux / Unix y Mac OSX, con el mismo aspecto y las funcionalidades exactas en todas las plataformas.(22)

La versión 0.15 no posee, al igual que CATIA y SolidWork, de un acelerador gráfico para el modelado y cálculo de los resortes helicoidales. El procedimiento en esta versión sería realizar una hélice en el módulo "*Part*" (Fig. A.7 del Anexo A). Posteriormente, se pasa a trabajar en el módulo "*Part Design*", se crea un plano y el perfil del alambre en el mismo; para ubicar este en el inicio de la hélice se puede desplazar con las restricciones (Fig. A.8 del Anexo A). Finalmente, con la herramienta "*Sweep*" del módulo "*Part*" se selecciona el camino y el perfil, se marca "crear sólido" y al aplicar queda conformado el resorte (Fig. A.9 del Anexo A).

Teniendo en cuenta lo antes descrito se puede evidenciar que, al igual que en CATIA y SolidWorks, es compleja la realización de los resortes de torsión y tracción.

[5] La Licencia Pública General Reducida (LGPL, pos sus siglas en inglés) garantiza la libertad de compartir y modificar el software cubierto por ella, asegurando que el software es libre para todos sus usuarios.(21)

1.3.2 Herramientas para el cálculo de resortes helicoidales.

Con el objetivo de agilizar el trabajo de diseño se han desarrollado distintas herramientas que automatizan el proceso de cálculo de las propiedades de los resortes helicoidales. Algunas de estas herramientas son: MITCalc, Springulator y eFunda.

1.3.2.1 MITCalc

MITCalc brinda un conjunto de cálculos de ingeniería, industriales, y técnicos para las rutinas del día a día de forma fiable y precisa.(23)

Esta herramienta brinda, además de los cálculos, las verificaciones comunes de diseño de disímiles elementos mecánicos como: engranajes rectos, engranajes cónicos, engranaje helicoidal, engranaje planetario, correa de distribución, cojinetes, resortes, viga, pandeo, placas, conchas, y muchos otros.(23)

Permite la salida directa a disímiles sistemas CAD propietarios, como:(23)

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

En la versión 1.19 de esta herramienta para los resortes helicoidales al introducir los datos iniciales: carga, material, la longitud precargada, entre otros (Fig. A.13 del Anexo A), se chequean los parámetros y aquellos que no satisfacen con un diseño factible son señalados (Fig. A.14 del Anexo A).

1.3.2.2 Springulator

Springulator es una herramienta realizada por Newcomb Spring Corp., empresa manufacturera de resortes de los Estados Unidos de América, que consta de dos aplicaciones para Android e iOS y un sitio web, que de manera sencilla y rápida, se realizan cálculos de diseño para los resortes helicoidales de compresión, torsión y tracción.

El mismo posee como valor agregado la funcionalidad de enviar el diseño a una dirección de correo especificada o a la empresa para ser contactado por un funcionario de la misma. Se caracteriza por poseer la cualidad de calcular a partir de la menor cantidad de datos iniciales las propiedades del resorte, algunas de las cuales son: esfuerzo, tensión, cantidad

de espiras, deflexión, entre otras. Además, muestra una gráfica del límite del esfuerzo y el esfuerzo corregido (Fig. A.15 del Anexo A).

1.3.2.3 eFunda

eFunda es un sitio web que posee como objetivo: crear un destino en línea para la comunidad de ingeniería, donde pueda encontrarse de forma concisa y confiable la mayoría de los cálculos que necesiten.(24)

eFunda brinda los conceptos básicos, es decir, cubre el material de nivel universitario de las escuelas de ingeniería. También, abarca exactamente en qué condiciones se aplican las fórmulas de esta rama y de los sustentos matemáticos relacionados a la misma. (24)

Uno de sus valores agregados es que brinda soporte a una serie de calculadoras *on-line* de finanzas, matemática, mecánica y diseño. Dentro de estas se encuentran un conjunto de calculadoras independientes para distintas propiedades de los resortes helicoidales, como: rango, deflexión, fuerzas, esfuerzos, fatiga, entre otras. Además, brinda una que agrupa todas las propiedades antes mencionadas (Fig. A.16 del Anexo A).

1.4 Metodologías de desarrollo

“La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado.”(25)

Algunas de las ventajas de utilizar una metodología son(25):

- Facilitar las tareas de planificación.
- Optimizar el uso de los recursos disponibles.
- Optimizar el conjunto y cada una de las fases del proceso de desarrollo
- Permitir la reutilización de partes del producto
- Confianza en los plazos de tiempo fijados en la definición del proyecto

En este apartado se realiza un estudio de algunas de las metodologías con el objetivo de seleccionar la que más se ajuste a las características de desarrollo del componente.

1.4.1 RUP

Proceso Racional Unificado (RUP, pos sus siglas en inglés) es una metodología basada en componentes, lo cual quiere decir, que el sistema en construcción está formado por componentes interconectados a través de interfaces bien definidas. Las tres frases claves

de esta metodología son: dirigida por casos de uso[6], centrada en la arquitectura, e iterativa e incremental. (27)

Dirigido por caso de uso significa que el proceso de desarrollo sigue un hilo, avanza a través de una serie de flujos de trabajo que parten de los casos de uso. Estos se especifican, se diseñan, y los casos de uso finales son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de prueba.(27)

Centrado en la arquitectura expresa que el proceso gira alrededor de la arquitectura seleccionada, la cual debe evolucionar en paralelo junto a los casos de uso. Esta abarca consideraciones de implementación, plataforma, subsistemas, clases, y componentes, entre otros aspectos de desarrollo que van apareciendo al especificarse y madurar los casos de uso.(27)

“El desarrollo de un producto de software comercial supone un gran esfuerzo que puede durar varios meses hasta posiblemente un año a más. Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y el incremento, al crecimiento del producto al avanzar las mismas.” (27)

Esta metodología conlleva un gran número de artefactos, está concebida para proyectos de larga duración y no posee un diseño simple, pero brinda procesos bien controlados.

1.4.2 SCRUM

“Scrum es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo del software. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora una nueva funcionalidad. Las iteraciones en general tienen una duración entre 2 y 4 semanas. Scrum se utiliza como marco para otras prácticas de ingeniería de software como RUP o eXtreme Programming.”(28)

“Se focaliza en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de inversión. Está diseñado especialmente para adaptarse a los cambios en los requisitos, por ejemplo en un mercado de alta competitividad. Los requisitos y las prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares.”(28)

[6] Un caso de uso es una unidad coherente de funcionalidad, expresada como transacción entre los actores y el sistema.(26)

“Scrum tiene un conjunto de reglas muy pequeño y muy simple y está basado en los principios de inspección continua, adaptación, auto-gestión e innovación. El cliente se entusiasma y se compromete con el proyecto dado que ve crecer el producto iteración a iteración y encuentra las herramientas para alinear el desarrollo con los objetivos de negocio de su empresa.”(28)

1.4.3 eXtreme Programming

Programación Extrema (XP, por sus siglas en inglés) “es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.” (29)
Algunas de sus características son(28):

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

frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

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

1.4.4 AUP

"El Proceso Unificado Ágil (AUP, por sus siglas en inglés) es un enfoque de modelado híbrido creado por Scott Ambler cuando combinó RUP con los métodos ágiles (A.M., por sus siglas en inglés). Mediante la combinación de RUP con AM, Ambler creó un marco sólido de procesos que se puede aplicar a todo tipo de proyectos de software, grandes o pequeños"(30).

Amber creó AUP bajo los siguientes principios(30):

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

A partir de esta metodología en la Universidad de las Ciencias Informáticas se define una versión que responda al proceso de desarrollo llevado a cabo en la institución. En esta versión se definen como fases de desarrollo: Inicio, Ejecución y Cierre (Tabla A.1 del Anexo A). Además, se proponen 11 roles en lugar de los 9 definidos por AUP. (Tabla A.2 del Anexo A)

Esta versión de AUP define cuatro escenarios en los que se puede ubicar el desarrollo de una aplicación de acuerdo a sus características, los cuales son(31):

- Escenario 1: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de

interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta.

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

Siguiendo la política de desarrollo de software de la institución, se define como metodología a emplear la AUP-UCI en el escenario número 4, debido a la necesidad de una metodología que responda con facilidad a los cambios continuos, por estar el cliente siempre acompañando el desarrollo y por estar bien definido el negocio.

1.5 Herramientas y lenguajes para el desarrollo

La acelerada evolución de las tecnologías informáticas ha llevado consigo un aumento exponencial de las herramientas y lenguajes que se emplean para el desarrollo de nuevos sistemas, que satisfagan las necesidades del hombre moderno; provocando que cada vez sea más necesario un estudio de estas a la hora de realizar un sistema determinado. Seguidamente se describen las características de las tecnologías empleadas en la presente investigación.

1.5.1 UML

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

modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código, así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos."(26)

"UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. La estructura estática define los tipos de objetos importantes para un sistema y para su implementación, así como las relaciones entre los objetos. El comportamiento dinámico define la historia de los objetos en el tiempo y la comunicación entre objetos para cumplir sus objetivos. El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos."(26)

"UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo, en un entorno de desarrollo complejo. Contiene construcciones para representar decisiones de implementación y para elementos de tiempo de ejecución en componentes."(26)

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

1.5.2 Visual Paradigm

Visual Paradigm es una herramienta CASE[7] que brinda un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. (32) Esta herramienta emplea el lenguaje de modelado UML.

Se caracteriza por(32):

- Disponibilidad en múltiples plataformas (Windows, Linux).

[7] Ingeniería de Software Asistida por Computadora

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

Se ha decidido su empleo para la confección de la propuesta de solución debido a que esta herramienta genera código en varios lenguajes, posee una licencia gratuita y emplea el lenguaje de modelado UML.

1.5.3 Open Cascade

La Tecnología Open Cascade (OCCT, por sus siglas en inglés), es una plataforma de desarrollo de software que proporciona servicios para superficies 3D y modelado de sólidos, el intercambio de datos CAD, y la visualización. La mayor parte de la funcionalidad OCCT se encuentra disponible en forma de bibliotecas de C ++. OCCT puede ser aplicado en el desarrollo de software cuyo objetivo sea el modelado 3D (CAD), fabricación / medición (CAM) o simulación numérica (CAE). Es una tecnología de software libre; se puede distribuir y/o modificar bajo los términos de la Licencia Pública General de GNU (LGPL) versión 2.1, con excepción adicional. (33)

Alternativamente, Open CASCADE puede ser utilizada bajo los términos de licencia comercial o acuerdo contractual. Está diseñada para ser altamente portátil y es conocida por trabajar en una amplia gama de plataformas (UNIX, Linux, Windows, Mac OS X, Android). La versión (7.0.0.beta) está certificada oficialmente en las plataformas Windows (IA-32 y x86-64), Linux (x86-64), MAC OS X (x86-64) y Android (4.0.4 ARMv7).(33)

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

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

1.5.4 Lenguaje C++

"C++ es un lenguaje compilado, es decir, para correr un programa su código tiene que ser procesado por un compilador produciendo archivos objetos, los cuales son combinados por un vínculo a un ejecutable del programa. (…) El ejecutable es creado por una combinación específica de hardware y software; no es portable, dígase, desde Mac a Windows. Cuando se habla de la portabilidad de programas en C++, usualmente significa portabilidad del código fuente; en otras palabras, el código fuente puede ser compilado satisfactoriamente y ejecutado en una variedad de sistemas." (34)

"La librería estándar de C++ puede ser implementada en el propio C++ (…). Esto implica que C++ es suficientemente expresivo y eficiente para los sistemas de mayor demanda de tareas de programación." (34)

C++ es usado por millones de programadores en cada dominio de aplicación. Billones de líneas de C++ están actualmente desarrolladas. Muchos sistemas operativos poseen partes esenciales escritas en C++, como Windows, iOS, Linux, entre otros. Este lenguaje no fue diseñado con la computación numérica en mente, sin embargo, muchos programas de ingeniería, numéricos y científicos son realizados en C++. Este puede coexistir con código escrito en otro lenguaje. C++ es soportado por una variedad de librerías y conjuntos de herramientas, tales como Boost[8], QT, OpenCV[9], entre otras. (34)

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

1.5.5 Framework Qt

"Qt es un marco de desarrollo de aplicaciones multiplataforma basado en C++, dentro de las que se encuentran: Linux, OS X, Windows, VxWorks, QNX, Android, iOS, Blackberry, Sailfish OS y otras."(35)

[8] Librerías portables.
[9] Procesamiento de imágenes en tiempo real.

Qt está disponible bajo varias licencias: licencia comercial y para software libre con varias versiones de la GPL[10] y la LGPL.(35)

"Qt es mucho más que un conjunto de herramientas de Interfaces Gráficas de Usuario (GUI, por sus siglas en inglés). Proporciona módulos para el desarrollo multiplataforma en las áreas de redes, bases de datos, OpenGL[11], tecnologías web, sensores, protocolos de comunicación (Bluetooth, puertos serie), XML[12] y procesamiento JSON[13], impresión, la generación de PDF, y mucho más".(35)

El Entorno Integrado de Desarrollo recomendado para el empleo de este framework es el Qt Creator.

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

1.5.6 Sistemas gestores de bases de datos

A continuación se realiza un estudio de los sistemas gestores de bases de datos[14] con el objetivo de identificar el más adecuado para almacenar la información de los resortes helicoidales y los materiales normalizados.

1.5.6.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional. Su código fuente se encuentra disponible libremente. Este sistema requiere un servicio de servidor para su utilización. Es uno de los sistemas de gestión de bases de datos de código abierto más potente del mercado. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema, un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (41)

Algunas de las características que posee son(41):

[10] Licencia Pública General (GPL, por sus siglas de inglés) garantiza la libertad de compartir y modificar todas las versiones de un programa que se encuentre bajo esta.(36)

[11] OpenGL es un entorno de desarrollo para aplicaciones gráficas en 2D y 3D.(37)

[12] Lenguaje Extensible de Enmarcado (XML, por sus siglas en inglés) es un formato de texto simple y muy flexible.(38)

[13] Notación de Objetos de JavaScript (JSON, por sus siglas en inglés) es un formato de intercambio de datos ligero.(39)

[14] Un sistema gestor de base de datos es un sistema que tiene como responsabilidad continua mantener los datos entre el momento en que se almacena y posteriormente el momento en el que se requiera para la recuperación.(40)

- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.
- Es una base de datos 100% ACID[15]
- Integridad referencial
- Juegos de caracteres internacionales
- Regionalización por columna
- Control Concurrente Multi-version (MVCC, por sus siglas en inglés)
- Múltiples métodos de autentificación
- Acceso encriptado vía SSL

1.5.6.2 MySQL

MySQL es un sistema gestor de bases de datos muy conocido y ampliamente usado por su simplicidad y notable rendimiento. Este sistema requiere de un servicio de servidor para su utilización. Algunas de las características que posee son(42):

- Está optimizado para equipos de múltiples procesadores.
- Es muy destacable su velocidad de respuesta.
- Se puede utilizar como cliente-servidor o incrustado en aplicaciones.
- Cuenta con un rico conjunto de tipos de datos.
- Su administración se basa en usuarios y privilegios.
- Es altamente confiable en cuanto a estabilidad se refiere.
- Se tiene constancia de casos en los que maneja cincuenta millones de registros, sesenta mil tablas y cinco millones de columnas.
- No incluye características de objetos como tipos de datos estructurados definidos por el usuario, herencia etc.

1.5.4.3 SQLite

“SQLite es un sistema gestor de base de datos SQL incorporado. A diferencia de la mayoría de las otras bases de datos SQL, SQLite no requiere un proceso de servidor independiente. SQLite lee y escribe directamente en archivos de disco ordinarios. Una base de datos completa de SQL con varias tablas, índices, activadores y vistas, está contenida en un solo archivo de disco. El formato de archivo de base de datos es multiplataforma - se puede copiar libremente una base de datos entre sistemas de 32 bits y de 64 bits. Estas

[15] ACID es el acrónimo en inglés de las propiedades: Atomicidad, Consistencia, Aislamiento y Durabilidad.

características hacen a SQLite una opción popular como un formato de archivo de aplicación.” (43)

“El código para SQLite es de dominio público y por lo tanto es libre su uso para cualquier propósito, comercial o privado. SQLite es la base de datos de mayor despliegue en el mundo con más aplicaciones de las que podemos contar, incluyendo varios proyectos de alto perfil.”(43)

Se decide el empleo de SQLite por su independencia de un servidor para su funcionamiento y porque se puede emplear en sistemas de 32 y 64 bits sin complicaciones.

1.6 Conclusiones del capítulo

El estudio de las herramientas similares permitió identificar que las propietarias, Autodesk Inventor y Solid Edge, poseen módulos independientes dedicados a acelerar el proceso de diseño de los resortes helicoidales, sin embargo, CATIA, SolidWorks y las herramientas libres no los contienen (siendo difícil el modelado con primitivas y funciones de barrido) ni brindan los cálculos de sus propiedades.

Además, se determinó que las herramientas de cálculos existentes no posibilitan el modelado del resorte diseñado, a excepción del MITCalc que se comunica con algunos sistemas CAD propietarios existentes.

De acuerdo a lo investigado en el presente capítulo se decidió emplear el lenguaje de programación **C++03**, el *framework* de desarrollo **Qt 4.8.6**, el Entorno de Desarrollo Integrado **Qt Creator 3.0.1, SQLite** como sistema gestor de base de datos y la biblioteca **Open Cascade Community Edition (OCE) 0.17**. Como metodología de desarrollo se decidió emplear **AUP-UCI** en el escenario 4 por las características que esta brinda y su acoplamiento con el trabajo que se realiza. El lenguaje de modelado seleccionado fue **UML 2.1** y como herramienta CASE el **Visual Paradigm para UML 8.0**.

2. Propuesta de solución

En el presente capítulo se aborda la propuesta de solución que incluye un modelo del dominio, la descripción del componente, los requisitos funcionales y no funcionales, las historias de usuario y los aspectos de diseño, como: el estilo y patrón arquitectónico, los patrones de diseño, el modelo de clases y diagramas de secuencias y de diseño de la base de dato. Además, se expone la metodología de cálculo a emplear para las propiedades de los resortes en la propuesta de solución.

2.1 Modelo del dominio

Un modelo de dominio facilita la comprensión de los sistemas debido a que permite clasificar y agrupar objetos, disminuyendo la complejidad y el número de elementos en el modelado. Además, se utiliza para escribir los objetos y sus relaciones con los sistemas, facilitando una mejor comunicación entre usuarios, desarrolladores y clientes al establecer un lenguaje común para el entendimiento del mismo.(44)

A continuación se presenta el modelo del dominio definido en la presente investigación:

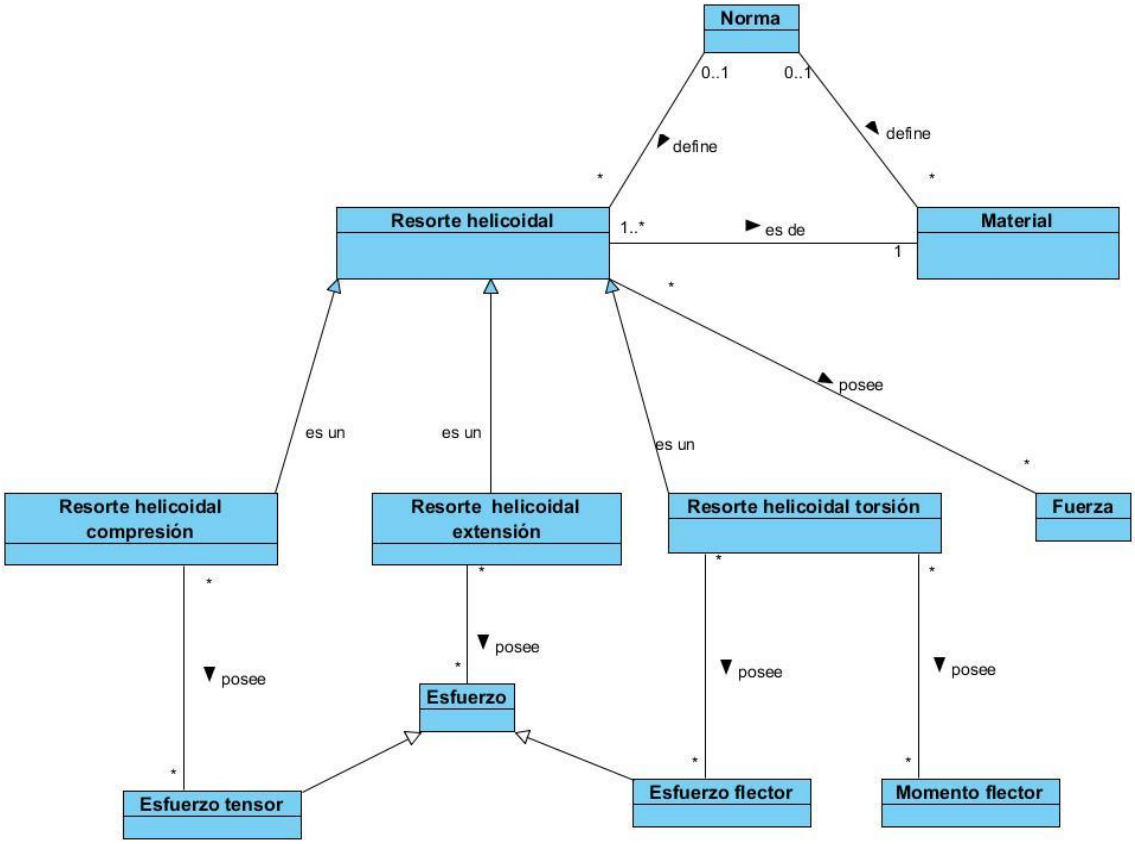


Fig. 7 Modelo del dominio

2.2 Definición de las clases del Modelo de dominio

Norma: es una especificación que define las propiedades de los elementos mecánicos por una agencia normalizadora o fabricante.

Resorte helicoidal: es un resorte en el que su cuerpo está caracterizado por la curva de una hélice.

Resorte helicoidal de compresión: es un resorte helicoidal que está especialmente diseñado para soportar fuerzas de compresión.

Resorte helicoidal de extensión: es un resorte helicoidal que soporta exclusivamente fuerzas de tracción y se caracteriza por tener un gancho en cada uno de sus extremos.

Resorte helicoidal de torsión: es un resorte helicoidal que trabaja a torsión o girando.

Momento flector: es la respuesta de efectuar una fuerza externa a la estructura ocasionando la flexión del mismo. “Se puede definir como la suma algebraica de los momentos de las fuerzas, a la derecha y a la izquierda de la sección.”(45)

Material: se asume como una variable que encapsula las propiedades de lo que está confeccionado algún objeto.

Fuerza: “descripción cualitativa de la interacción entre dos cuerpos o entre un cuerpo y su entorno”(46) “En el lenguaje cotidiano, una fuerza es un empuje o un jalón. Para medir tales fuerzas en forma cuantitativa, la expresamos en términos de aceleración que determinado cuerpo estándar experimenta en respuesta a esa fuerza.”(47)

Esfuerzo: “caracteriza la intensidad de las fuerzas que causan el cambio de forma, generalmente con base en la fuerza por unidad de área.”(46)

Esfuerzo flector: es la relación entre el momento flector y el momento de inercia ($-\frac{My}{I}$) (10)

Esfuerzo tensor: es el esfuerzo obtenido de aplicar una fuerza perpendicular a una sección de área transversal determinada.(46)

2.3 Descripción del componente

El módulo será diseñado con las funcionalidades necesarias para modelar geométricamente resortes helicoidales de varias clases (compresión, extensión y torsión) y realizará los cálculos asociados a su diseño en correspondencia con las normas establecidas y otras fuentes literarias.

La interfaz poseerá una ventana principal con tres pestañas, una para cada tipo de resorte, las que al activarse muestren los campos para introducir los datos iniciales; en la parte superior de cada una de las tres vistas habrá una ilustración que representa un resorte con los datos fundamentales, la ilustración del tipo de compresión se corresponderá con el tipo de extremo seleccionado. Las vistas de cada pestaña poseerán, además, un botón "*Standards*" que tendrá asociada una caja de diálogo para cargar tipos de resortes normalizados, que se encuentran almacenados en una base de datos SQLite.

La caja de diálogo de resortes estándares poseerá un árbol de visualización en el que se representen los nombres de los componentes normalizados de la base de datos; en el caso del de compresión, las formas de los extremos se designarán como un subgrupo de los nombres de los componentes, en el de torsión se agruparán por ángulo de inclinación de los extremos, y en el de extensión sólo se representarán los nombres de la norma.

Las tres vistas de las pestañas poseerán un botón "*Cancel"* que cierra la ventana y un botón "*Calculate"*, el cual muestre la ventana con los resultados de los cálculos correspondientes a la pestaña activa.

Las ventanas de cálculos poseerán grupos de elementos que agrupen las propiedades, como las dimensiones, cargas, fatiga, entre otros. También, contendrá una tabla de resultados en la que se muestran otras características con los valores correspondientes. En la sección de materiales se encontrará una caja de verificación la cual tendrá asociada una ventana en la que se cargan datos de materiales de la base de datos. La vista de los materiales normalizados contendrá un árbol de visualización en el que se muestran los nombres de las normas, una tabla que se relaciona con la norma seleccionada (al dar doble clic sobre una de sus filas se debe cerrar la ventana y actualizarse los datos del material en la vista anterior), y un botón "*Cancel"* para cerrar la ventana. En los campos de la fatiga se encontrará una caja de verificación que indicará si se tendrá en cuenta la fatiga para el diseño, permitiéndose así la inserción de los datos en dicha área. En la parte inferior se encontrará una sección en la que se muestran mensajes de errores de diseño. La ventana poseerá dos botones, uno de cancelar ("*Cancel"*) que cierre la misma y otro de dibujar ("*Draw"*) que la cierre y modele el resorte en el visor.

2.4 Requisitos

"Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un

dispositivo, hacer un pedido o encontrar información."(48) Se pueden clasificar en funcionales y no funcionales. Existen diferentes métodos para la descripción de los requisitos, uno de ellos es el uso de las Historias de Usuario.

2.4.1 Requisitos Funcionales

Los requisitos funcionales son "las declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a las entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer".(48)

A continuación se enumeran los requisitos funcionales definidos para el componente:

RF 1. Insertar parámetros del resorte helicoidal de compresión.

RF 2. Insertar parámetros del resorte helicoidal de extensión.

RF 3. Insertar parámetros del resorte helicoidal de torsión.

RF 4. Seleccionar un resorte helicoidal de compresión por una norma determinada.

RF 5. Seleccionar un resorte helicoidal de extensión por una norma determinada.

RF 6. Seleccionar un resorte helicoidal de torsión por una norma determinada.

RF 7. Mostrar los resultados del cálculo de las propiedades del resorte helicoidal de compresión.

RF 8. Mostrar los resultados del cálculo de las propiedades del resorte helicoidal de extensión.

RF 9. Mostrar los resultados del cálculo de las propiedades del resorte helicoidal de torsión.

RF 10. Actualizar parámetros del material.

RF 11. Seleccionar un material normalizado por una norma determinada.

RF 12. Insertar parámetros para la fatiga.

RF 13. Mostrar los errores de diseño.

RF 14. Modelar el resorte helicoidal diseñado.

2.4.2 Requisitos No Funcionales

Los requisitos no funcionales son "restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema".(48)

La taxonomía de los requisitos no funcionales empleada en la investigación es la definida por la metodología AUP-UCI.

A continuación se exponen los requisitos no funcionales del componente:

Software:

SO.: GNU-Linux, GCC: 4.2.4

Hardware:

RAM: 1 Gb, Espacio libre en disco: 1 Gb.

Funcionalidad:

Precisión: debe poseer una alta precisión, en cuanto a la cantidad de cifras significativas, en los datos introducidos y mostrados.

Usabilidad:

Comprensibilidad: al pasar el cursor sobre la variable se debe mostrar la propiedad a la que representa, así como imágenes que apoyen la compresión de las mismas.

2.4.3 Historias de usuario

Las historias de usuario son tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento se pueden modificar, reemplazar por otras más específicas o generales o añadirse nuevas. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.(49)

A continuación se muestra una de las historias de usuario realizadas en la propuesta de solución. Para el estudio de las demás historias de usuario remitirse al Anexo B.

Tabla 1 Historia de Usuario "Insertar parámetros de entrada del resorte helicoidal de compresión".

Número: 1	**Nombre del requisito:** Insertar parámetros del resorte helicoidal de compresión.
Programador: Gustavo García González	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 2 semanas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 1,5 semanas
Descripción: 1. Objetivo:	

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

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

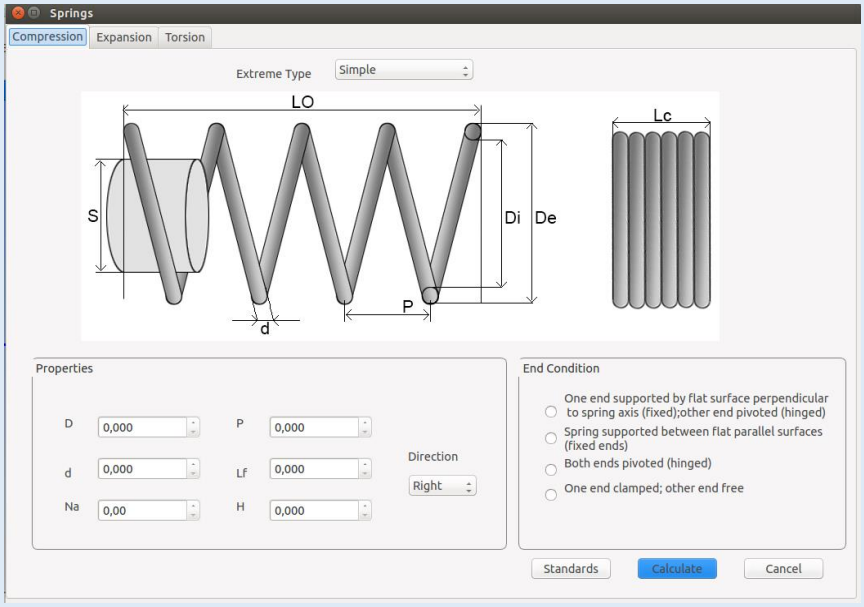
Para insertar parámetros del resorte helicoidal de compresión hay que:

- Tener en cuenta los siguientes datos: diámetro del alambre, diámetro medio, longitud bajo carga, cantidad de espiras, paso entre espiras, longitud de trabajo, tipo de extremo, dirección de la hélice y condición de los extremos.

3. Comportamiento válido y no válido:
   - El diámetro del alambre, diámetro medio, longitud bajo carga, cantidad de espiras, paso entre espiras y la longitud de trabajo deben ser valores numéricos reales positivos y son campos obligatorios.
   - La dirección de la hélice: lista desplegable.
   - Los tipos de extremos: lista desplegable.
   - Sólo se debe escoger una de las condiciones de los extremos.
   - El diámetro del alambre debe ser menor o igual que el paso y menor que el diámetro medio.
   - La longitud libre del resorte menos la longitud de carga debe ser mayor o igual a la longitud de trabajo del resorte.
4. Flujo de acción:
   - Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción "*Calculate*" se crea el resorte, se calculan las restantes propiedades y se muestra la vista de los resultados de los cálculos.
   - Si se selecciona el botón "*Cancel*" se cierra la ventana.
   - Si se selecciona el botón "*Standards*" se muestra la ventana con los datos de los estándares.
   - Si los datos están incorrectos se señalarán los campos en cuestión dando la posibilidad al usuario de realizar nuevamente la acción en cuestión.

Observaciones:

Prototipo de interfaz:



2.5 Diseño

“La esencia del diseño del software es la toma de decisiones sobre la organización lógica del software. Algunas veces, se representa esta organización lógica como un modelo en un lenguaje definido de modelado tal como UML y otras veces simplemente se utiliza notaciones informales y esbozos para representar el diseño”.(48)

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

2.5.1 Estilo y patrón arquitectónico del software.

“Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una estructura para todos los componentes del sistema. En caso de que una arquitectura existente se vaya a someter a reingeniería, la imposición de un estilo arquitectónico desembocará en cambios fundamentales en la estructura del software, incluida una reasignación de la funcionalidad de los componentes.” (50)

Se escoge como estilo arquitectónico el de Llamada y Retorno, pues “permite que un diseñador de software obtenga una estructura de programa que resulta relativamente fácil modificar y cambiar de tamaño”. (50)

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

“Un patrón arquitectónico define un enfoque específico para el manejo de alguna característica de comportamiento del sistema”. (50)

“Los patrones se usan junto con un estilo arquitectónico para determinar la forma de la estructura general de un sistema”. (50)

2.5.1.1 Arquitectura en Capas

El modelo en capas organiza el sistema en capas, cada una de las cuales proporciona un conjunto de servicios a la capa superior. Este modelo soporta el desarrollo incremental del sistema. También, soporta bien los cambios y es portable. Además, cuando las interfaces de las capas cambian o se añaden nuevas funcionalidades a una capa, solamente se ven afectadas las capas adyacentes. Entre las desventajas de este modelo es que la estructuración de los sistemas puede resultar difícil y el rendimiento se puede ver afectado,

pues se puede incurrir en que una funcionalidad debe pasar por muchas capas para alcanzar la capa superior que solicitó su servicio. (37)
Se decide el empleo de este modelo en la confección del componente porque soporta bien los cambios y el desarrollo incremental.

2.5.2 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. La propuesta de solución contendrá los siguientes patrones de diseño:

- Experto: mediante su uso, se asignan responsabilidades a la clase que cuenta con la información necesaria. Se evidenciará en la clase "*compressionCalculation*", pues esta poseerá un objeto de la clase "*compressionSprings*" para poder acceder a la información y funcionalidades de los resortes de compresión.
- Creador: permite crear objetos de una clase determinada. Se utilizará en algunas de las clases de interfaz para crear instancias de formularios y entidades.
- Polimorfismo: se emplea para asignar comportamientos distintos según el tipo de clase. Se evidenciará en las clases "*compresionSpring*", "*torsionSpring*" y "*extensionSpring*", que heredarán de la clase abstracta "*springs*" y definirán el comportamiento de la funcionalidad "*makeSpring*".
- Alta cohesión: este patrón caracteriza a las clases que posean responsabilidades estrechamente relacionadas, es decir, que no realicen un trabajo enorme. Con el objetivo de que las clases "*standardsSpringsDlg*" y "*standardsMaterialDlg*" no realizaran un trabajo enorme y poder reutilizar código se creará la clase "*dbconnection*", encargada de las funcionalidades de conexión a la base de datos.
- Bajo acoplamiento: posibilita que una clase no dependa mucho de otras clases. Este patrón se empleará en la clase "*standardsSpringsDlg*" y "*standardsMaterialDlg*", pues al crear la clase "*dbconnection*" se disminuye la dependencia al no tener que conocer ni recurrir demasiado a la clase que brinda el Qt de conexión a la base de datos.
- Fabricación pura: permite asignar un conjunto altamente cohesivo de responsabilidades a una clase artificial. Se utilizará en la clase "*dbconnection*",

la cual será creada para asegurar la reutilización del código, la cohesión y el bajo acoplamiento en las clases "*standardsSpringsDlg*" y "*standardsMaterialDlg*".

2.5.3 Diagrama de clase del diseño

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

A continuación se muestra el diagrama de clase diseñado de la solución:

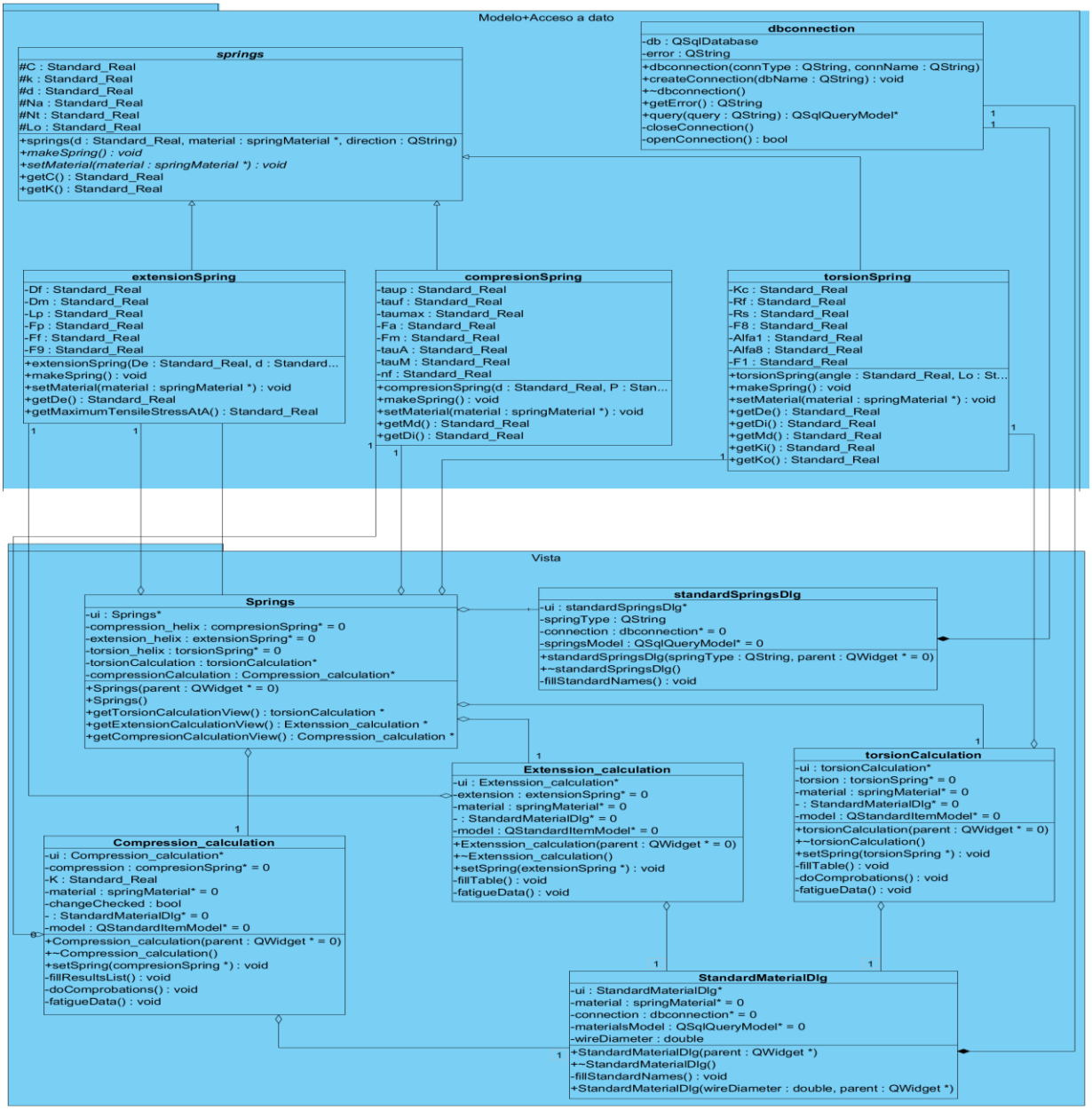


Fig. 8 Diagrama de clase del diseño (síntesis)

2.5.4 Diagramas de secuencia del diseño

Un diagrama de secuencia muestra un conjunto de mensajes, dispuestos en una secuencia temporal; puede mostrar un escenario, es decir, una historia individual de una transacción. Uno de sus usos es mostrar la secuencia del comportamiento de un caso de uso. Cuando está implementado el comportamiento, cada mensaje en un diagrama de secuencia corresponde a una operación en una clase, a un evento disparador, o a una transición en una máquina de estados.(26)

A continuación se presenta el diagrama de secuencia correspondiente a la historia de usuario Insertar parámetros del resorte helicoidal de compresión. Para el estudio de los demás diagramas de secuencia del diseño remitirse al Anexo B.

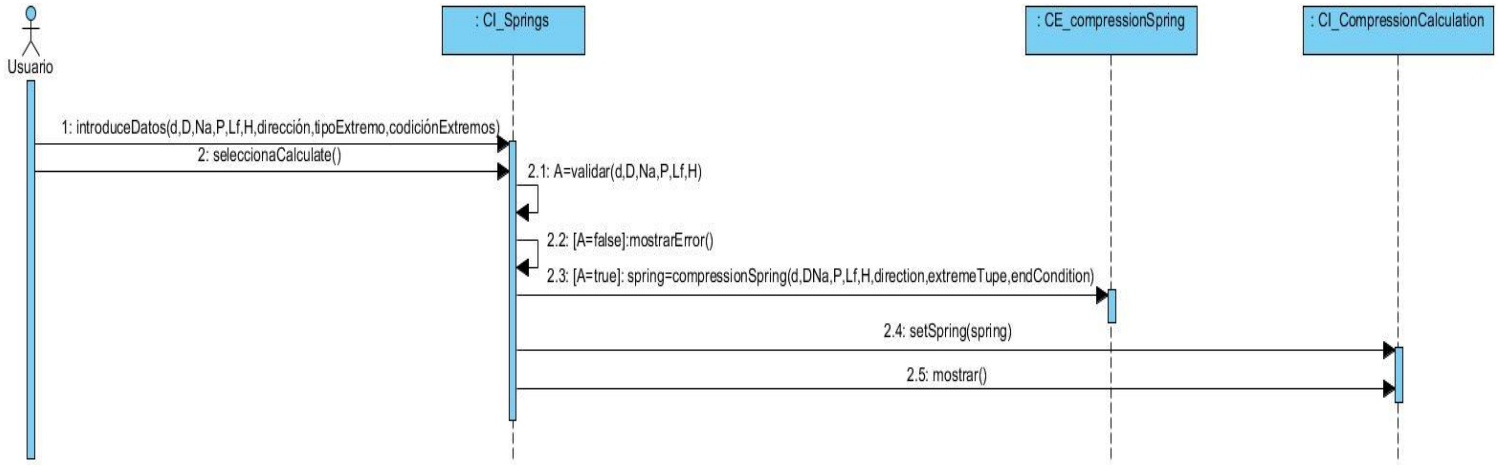


Fig. 9 Diagrama de Secuencia de la Historia de Usuario "Insertar parámetros del resorte helicoidal de compresión".

2.5.5 Diseño de la base de datos

"El diseño de una base de datos consiste en definir la estructura de los datos que debe tener la base de datos de un sistema de información determinado. En el caso relacional, esta estructura será un conjunto de esquemas de relación con sus atributos, dominios de atributos, claves primarias, claves foráneas, etc.". (51)

"El modelo entidad-relación es uno de los enfoques de modelización de datos que más se utiliza actualmente por su simplicidad y legibilidad. Su legibilidad se ve favorecida porque proporciona una notación diagramática muy comprensiva. Es una herramienta útil tanto para ayudar al diseñador a reflejar en un modelo conceptual los requisitos del mundo real de interés como para comunicarse con el usuario final sobre el modelo conceptual obtenido y, de este modo, poder verificar si satisface sus requisitos".(51)

A continuación se muestra el diagrama Entidad-Relación diseñado para almacenar los estándares de los resortes helicoidales y los materiales en la base de datos:

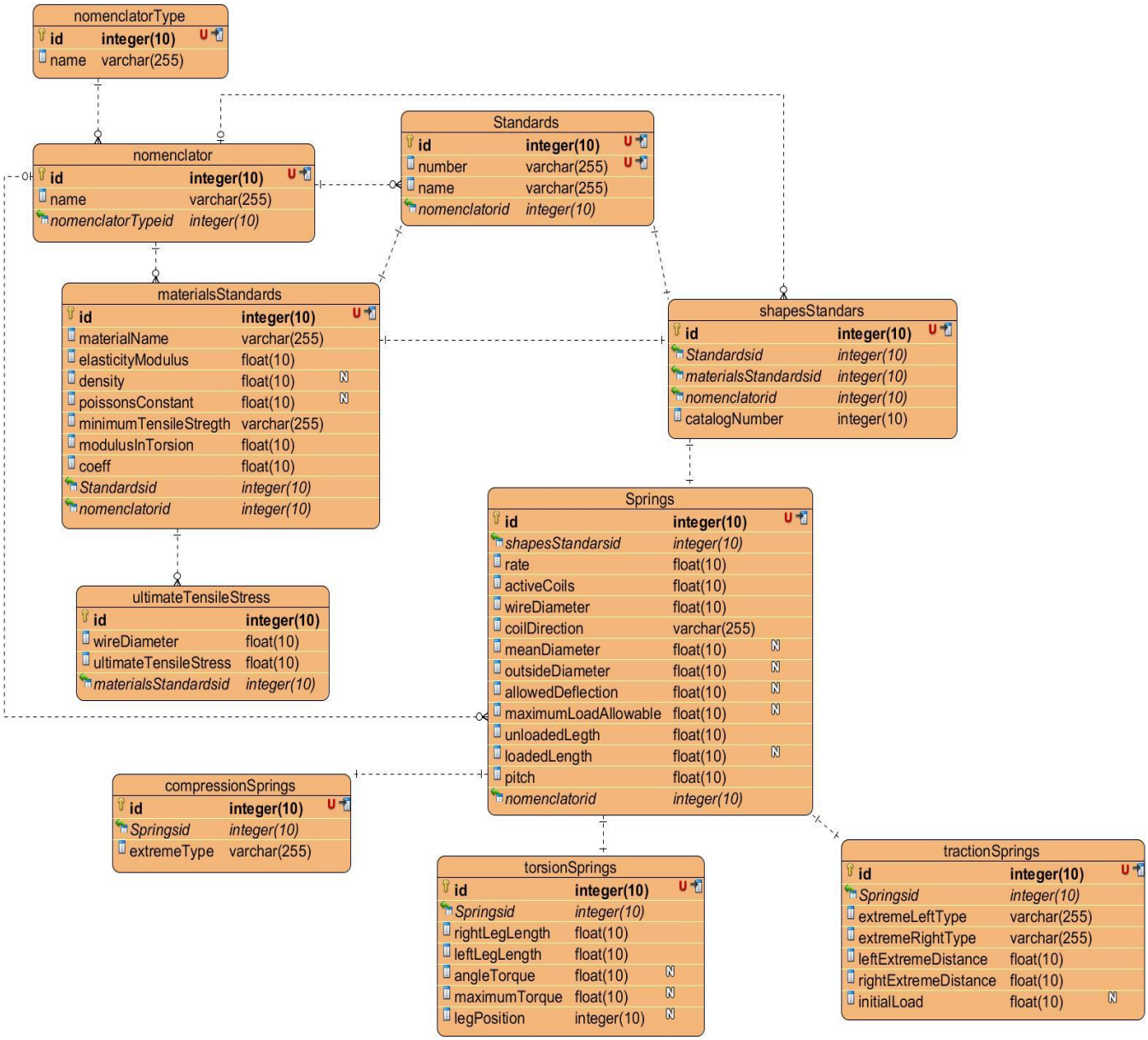


Fig. 10 Modelo Entidad-Relación de la base de datos.

2.5.5.1 Descripción de las tablas

En esta sección se presenta una breve descripción de cada uno de los atributos de la tabla "*Springs*", el resto de las descripciones de las tablas de la base de datos se encuentran en el Anexo B.

Tabla 2 Descripción de la tabla de la BD "Springs"

Springs		
Descripción: En la presente tabla se muestran los datos comunes de los resortes estandarizados.		
Atributo	Tipo	Descripción
Id	integer(10)	Etiqueta única que identifica al resorte.
rate	float(10)	Almacena el índice del resorte.
activeCoils	float(10)	Almacena la cantidad de espiras activas del resorte
wireDiameter	float(10)	Almacena el diámetro del alambre del resorte.
coilDirection	varchar(255)	Almacena la dirección de la hélice del resorte.
meanDiameter	float(10)	Almacena el diámetro medio del resorte.
outsideDiameter	float(10)	Almacena el diámetro exterior del resorte.
allowedDeflection	float(10)	Almacena la deflexión permitida del resorte.

maximumLoadAllowable	float(10)	Almacena la carga máxima permitida del resorte.
unloadedLegth	float(10)	Almacena la longitud libre del resorte.
loadedLength	float(10)	Almacena la longitud cargada permitida del resorte.
pitch	float(10)	Almacena la longitud del paso entre las espiras del resorte.
shapesStandardsid	integer(10)	Almacena el identificador único de la tabla *shapesStandards*
nomenclatorid	integer(10)	Almacena el identificador único de la tabla *nomenclator*

2.6 Metodología de cálculo

2.6.1 Metodología de cálculo para resortes helicoidales de compresión

En esta metodología de cálculo se definieron los siguientes datos iniciales: diámetro del alambre(d), diámetro medio(D), cantidad de espiras activas(Na), paso o distancia entre espiras(P), longitud bajo carga(Lf) y longitud de trabajo (H).

Tabla 3 Metodología de Cálculo para los resortes helicoidales de compresión.

No.	Fórmula	Propósito
1	n=Na (extremo simple) n=Na+2 (extremo escuadrado y escuadrado amolado) n=Na+1 (extremo simple amolado)	Calcular la cantidad total de espiras.(10)
2	Lo=P*n+d (extremo simple) Lo=P*Na+3*d (extremo escuadrado) Lo=P*n (extremo simple amolado) Lo= P*Na*+2*d (extremo escuadrado amolado)	Calcular la longitud total del resorte libre.(10)
3	Ls=d*(n+1) (extremo simple y escuadrado) Ls=d*n (extremo simple amolado y escuadrado amolado)	Calcular la longitud sólida del resorte.(10)

4	Di=D-d	Calcular el diámetro interior.
5	De=D+d	Calcular el diámetro exterior.
6	C=D/d	Calcular el índice del resorte.
7	kb=(4*C+2)/(4*C-3) kw=((4*C-1)/(4*C-4))+(0.615/C) ks=(2*C+1) / (2*C) kc=kb/ks	Calcular las correcciones de la curva por los distintos criterios.(10)
8	k=(d⁴)*G/(8*(D³)*Na)	Calcular la constante de diseño.(30)
9	Lp=Lf+H	Calcular la longitud de precarga.
10	Dp=Lo-Lp	Calcular la deflexión de la precarga.
11	Dm=Lo-Ls	Calcular la deflexión máxima.
12	Fs=k*Dm	Calcular la fuerza para alcanzar la longitud sólida.(10)
13	Fop=0.75*Fs	Calcular la fuerza de operación.(10)
14	Fmax=0.85*Fs	Calcular la fuerza máxima.(10)
15	Fp=k*Dp	Calcular la fuerza de precarga.
16	Ff=k*(Lo-Lf)	Calcular la fuerza de carga.
17	l=π*D*Nt	Calcular la longitud del alambre.
18	W8=(Lo-Lf)²*k/2/1000	Calcular la energía de deformación.
19	f=√(G/(2*ρ))*d/1000000/(2*π*Na*D²)	Calcular la frecuencia crítica.
20	τ1=8*Fp*D/(π*d³)*K_empleado	Calcular el esfuerzo en la precarga.(52)
21	τ8=8*Ff*D/(π*d³)*K_empleado	Calcular el esfuerzo en la carga aplicada.
22	τ9=8*Fs*D/(π*d³)*K_empleado	Calcular el esfuerzo en la carga máxima.
23	V=1000*(τ9-τ8)/√(2*G*ρ)	Calcular la velocidad crítica.
24	n=τa/τ8	Calcular el valor de seguridad sin tener en cuenta la fatiga.
25	mass= π*d²*ρ*l/4/1000000000	Calcular la masa.
26	Fa=(Ff-Fp)/2	Calcular la fuerza alternante(10)
27	Fm=(Ff+Fp)/2	Calcular la fuerza promedio(10)
28	τA=K_empleado*8*Fa*Md/(π*d³)	Calcular el esfuerzo de la fuerza alternante(10)
29	τM=K_empleado*8*Fm*Md/(π*d³)	Calcular el esfuerzo de la fuerza promedio(10)
30	Ssu=0.67*Sut	Calcular el módulo de ruptura torsional(10)
31	Sse=τA/(1-(τM)/Sse)	Calcular el intercepto de la línea de fatiga(Ver epígrafe 1.3.3.1)
32	Se=Sse/0.577	Calcular el límite de la fatiga(Ver epígrafe 1.3.3.1)
33	R=τA/τM	Calcular la pendiente de la línea de carga(10)
34	Ssa=R*Sse*Ssu/(R*Ssu+S)	Calcular la componente de amplitud de la fuerza(10)
35	n=Ssa/τA	Calcular el nivel de seguridad(10)
36	a=(f*Sut)²/Se	Calcular constante a para la fatiga(10)
37	b=-log10(f*Sut/Se)/3	Calcular constante b para la fatiga(10)
38	Sflow=Sut*N^(log10(f)/3)	Calcular la resistencia a la fatiga con una cantidad de oscilaciones pequeñas(Ver epígrafe 1.3.3.1)

39	Sfheigth=Se	Calcular la resistencia a la fatiga con una cantidad de oscilaciones grandes(Ver epígrafe 1.3.3.1)
40	Sfmedium=a*N^b	Calcular la resistencia a la fatiga con una cantidad de oscilaciones media(Ver epígrafe 1.3.3.1)

2.6.2 Metodología de cálculo para resortes helicoidales de extensión.

En esta metodología de cálculo se definieron los siguientes datos iniciales: diámetro del alambre (d), diámetro exterior (De), distancia entre el cuerpo y el gancho izquierdo (D1), distancia entre el cuerpo y el gancho derecho (D2), longitud libre (Lo), longitud bajo carga(Lf), longitud de trabajo(H), paso o distancia entre espiras(P), y fuerza inicial (Fi).

Tabla 4 Metodología de cálculo para los resortes helicoidales de extensión.

No.	Fórmula	Propósito
1	Lh=Lo-D1-D2	Calcular la longitud del cuerpo.
2	Nb=(Lh-d)/P	Calcular la cantidad de espiras del cuerpo.
3	D=De-d	Calcular el diámetro medio.
4	k=(d^4)*G/(8*(D^3)*Na)	Calcular la constante de diseño.(10)
5	r1=De/2-d/2	Calcular el radio de la hélice.
6	r2=2*d	Calcular el radio de transición entre el cuerpo y el extremo.
7	C=D/d	Calcular el índice del resorte.(10)
8	C1=2*r1/d	Calcular el índice del extremo.(10)
9	C2=2*r2/d	Calcular el índice de la transición entre el cuerpo y el extremo.(10)
10	(K)a=(4*$C1^2$ -C1-1)/(4*C1*(C1-1))	Calcular el factor de corrección de la curva en el extremo.(10)
11	(K)b=(4*C2-1)/(4*C2-4)	Calcular el factor de corrección de la curva en la transición.(10)
12	kb=(4*C+2)/(4*C-3) kw=((4*C-1)/(4*C-4))+(0.615/C) ks=(2*C+1) / (2*C) kc=kb/ks	Calcular las correcciones de la curva por los distintos criterios.(10)
13	Do=Fi/k	Calcular la deflexión de la fuerza inicial o de pretensado.
14	Li=Lo+Do	Calcular la longitud inicial o de pretensado.
15	Lp=Lf-H	Calcular la longitud de precarga.
16	Dp=Lp-Lo	Calcular la deflexión de la precarga.
17	Df=Lf-Lo	Calcular la deflexión de la carga.
18	Fp=k*Dp+Fi	Calcular la fuerza de precarga.
19	Ff=k*Df+Fi	Calcular la fuerza de la carga.
20	F=Fp+Ff+Fi	Calcular la fuerza total.
21	F9= π*τa*d^3/(8*D)/K_empleado	Calcular la fuerza máxima.
22	D9=(F9-Fi)/k	Calcular la deflexión máxima.
23	L9=Lo+D9	Calcular la longitud máxima.

24	(τ)a=F*(K)a*(16*D/(π*d³))+(4/(π*d²))	Calcular el esfuerzo máximo de tensión.(10)
25	(τ)b=(K)b*(8*F*D/(π*d³))	Calcular el esfuerzo máximo torsional.(10)
26	mass= π*d²*ρ*l/4/1000000000	Calcular la masa.
27	τ1=8*Fp*D/(π*d³)*K_empleado	Calcular el esfuerzo en la precarga.(52)
28	τ8=8*Ff*D/(π*d³)*K_empleado	Calcular el esfuerzo en la carga aplicada.
29	W8=(Ff+Fi)*Df/2000	Calcular la energía de deformación
30	f=√(G/(2*ρ))*d/1000000/(2*π*Na*D²)	Calcular la frecuencia crítica
31	n=τa/τ8	Calcular el factor de seguridad sin tener en cuenta la fatiga
33	Fa=(Ff-Fp)/2	Calcular la fuerza alternante(10)
34	Fm=(Ff+Fp)/2	Calcular la fuerza promedio(10)
35	τA=K_empleado*8*Fa*Md/(π*d³)	Calcular el esfuerzo de la fuerza alternante(10)
36	τM=K_empleado*8*Fm*Md/(π*d³)	Calcular el esfuerzo de la fuerza promedio(10)
37	Ssu=0.67*Sut	Calcular el módulo de ruptura torsional(10)
38	Sse=τA/(1-(τM)/Sse)	Calcular el intercepto de la línea de fatiga(Ver epígrafe 1.3.3.1)
39	Se=Sse/0.577	Calcular el límite de la fatiga(Ver epígrafe 1.3.3.1)
40	nbody=((τA/Sse)*pow(Ssu/τM,2)*(-1+√(1+(2*((tauM/Ssu)*(Sse/tauA))²)))/2	Calcular el nivel de seguridad en el cuerpo del resorte(10)
41	σA=Fa*(16*Md/(π*d³)+4/(π*d²))	Calcular el esfuerzo alternante en el gancho(10)
42	σM=Fm/Fa*σA	Calcular el esfuerzo promedio en el gancho(10)
43	nhook=((Sut/σM)²*σA/Se*(-1+√(1+(2*(σM/Sut)*(Se/σA))²)))/2	Calcular el nivel de seguridad en el gancho del resorte(10)
44	a=(f*Sut)²/Se	Calcular constante a para la fatiga(10)
45	b=-log10(f*Sut/Se)/3	Calcular constante b para la fatiga(10)
46	Sflow=Sut*N^(log10(f)/3)	Calcular la resistencia a la fatiga con una cantidad de oscilaciones pequeñas(Ver epígrafe 1.3.3.1)
47	Sfheigth=Se	Calcular la resistencia a la fatiga con una cantidad de oscilaciones grandes(Ver epígrafe 1.3.3.1)
48	Sfmedium=a*N^b	Calcular la resistencia a la fatiga con una cantidad de oscilaciones media(Ver epígrafe 1.3.3.1)

2.6.3 Metodología de cálculo para resortes helicoidales de torsión.

En esta metodología de cálculo se definieron los siguientes datos iniciales: ángulo entre los extremos (angle), diámetro exterior (De), posición de la carga(Rf), longitud del extremo

izquierdo (R1), longitud del extremo derecho (R2), fuerza aplicada (F8), ángulo de trabajo (αh), diámetro del alambre (d) y paso o distancia entre espiras(P).

Tabla 5 Metodología de cálculo para los resortes helicoidales de torsión.

No.	Fórmula	Propósito
1	Nt= (Lo-d)/P	Calcular la cantidad de espiras.
2	Di=De-2*d	Calcular el diámetro interior.
3	r=Di/2+d/2	Calcular el radio de la hélice.
4	D=De-d	Calcular el diámetro medio.
5	C=D/d	Calcular el índice del resorte.
6	Kg=(C/(C-1) + 1/(4*C) + 1/(16*C²))/(1+3/16*1/(C² – 1)) [Gohner] Ki=(4*C² -C-1)/(4*C*(C-)) [Wahl interior] Ko=(4*C² -C-1)/(4*C*(C-)) [Wahl exterior]	Calcular el factor de corrección del esfuerzo de la curva.(10) (53)
7	l= π*D*Nt*R1*R2	Calcular la longitud del alambre.
8	k=E*d⁴/(64*180/π)*Na*D)/1000	Calcular la constante del diseño.(10)
9	M8=F8*Rf/1000	Calcular el momento flector de la fuerza aplicada.(10)
10	α8=M8/k	Calcular la deflexión angular de la fuerza aplicada.
11	α1=α8-αh	Calcular la deflexión angular de la precarga.
12	M1=α1*k	Calcular el momento flector de la precarga.
13	F1=M1*1000/Rf	Calcular la fuerza de precarga.
14	M9= π* τa*d³/32*1000/K_empleado	Calcular el momento flector máximo.
15	F9=1000*M9/Rf	Calcular la fuerza máxima.
16	α9=1000*M9/k	Calcular la deflexión angular dela fuerza máxima.
17	W8= π*M8*α8/360	Calcular la energía de la deformación.
18	τ1=32*1000*M1/(π*d³)*K_empleado	Calcular el esfuerzo en la precarga.(10)
19	τ8=32*1000*M8/(π*d³)*K_empleado	Calcular el esfuerzo de la fuerza aplicada.
20	n=τa/τ8	Calcular el factor de seguridad sin tener en cuenta la fatiga.
21	mass= π*d²*ρ*l/4/1000000000	Calcular la masa.
22	Ma=(Mf-Mp)/2	Calcular el momento alternante(10)
23	Mm=(Mf+Mp)/2	Calcular el momento promedio(10)
24	τA=K_empleado*32*1000*Ma/(π*d³)	Calcular el esfuerzo alternante(10)
25	τA=K_empleado*Mm/Ma*τA	Calcular el esfuerzo promedio(10)
26	Ssu=0.67*Sut	Calcular el módulo de ruptura torsional(10)
27	Sse=τA/(1-(τM)/Sse)	Calcular el intercepto de la línea de fatiga(Ver epígrafe 1.3.3.1)
28	Se=Sse/0.577	Calcular el límite de la fatiga(Ver epígrafe 1.3.3.1)
29	n=((τA/Se)*pow(Sut/τM,2)*(-1+√(1+(2*(tauM/Sut)*(Se/tauA))²)))/2	Calcular el nivel de seguridad(10)
30	a=(f*Sut)²/Se	Calcular constante a para la fatiga(10)

31	b=-log10(f*Sut/Se)/3	Calcular constante b para la fatiga(10)
32	Sflow=Sut*N^(log10(f)/3)	Calcular la resistencia a la fatiga con una cantidad de oscilaciones pequeñas
33	Sfheigth=Se	Calcular la resistencia a la fatiga con una cantidad de oscilaciones grandes
34	Sfmedium=a*N^b	Calcular la resistencia a la fatiga con una cantidad de oscilaciones media

2.7 Conclusiones del capítulo

En este capítulo se identificó el estilo y patrón arquitectónico, los patrones de diseño y los requisitos, elementos fundamentales para la etapa de implementación. Además, se realizaron las historias de usuario y los prototipos de interfaz correspondientes a los requisitos del componente, brindando una descripción detallada de los mismos por parte del cliente. También, se generaron los diagramas de clase del diseño, secuencia y diseño de la base de datos, ofreciendo una mejor perspectiva a la hora de realizar la siguiente fase. Finalmente, se conformó la metodología de cálculo a emplear para cada uno de los tipos de resortes helicoidales.

3 Implementación y pruebas

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

3.1 Implementación

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

3.1.1 Estándar de codificación

Los estándares de codificación son pautas a cumplir por el código realizado en determinado sistema, con el objetivo de garantizar que todos los participantes lo puedan entender en menor tiempo y que el código en consecuencia sea mantenible, o sea, que sea fácil de modificar para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento.

A continuación se expone el estándar empleado en la solución.

Tabla 6 Estándar de codificación del componente (52).

Descripción	Ejemplo
Definición de Objetos, Clases, funciones y atributos	

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

Siempre se declara para todas las clases implementadas su respectivo destructor de clase.	virtual ~Foo()

La declaración de funciones o métodos siempre comenzarán en letra inicial minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase.	<Tipo dato retorno> funcion() <Tipo dato retorno> funcionCompuesta() <Tipo dato retorno> funcionDobleCompuesta()
Los atributos siempre estarán escritos con letra minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase.	<Tipo dato> atributo; <Tipo dato> atributoNombreCompuesto;
Definición de parámetros dentro de las funciones y constructores de clases	
Los nombres de los identificadores de los parámetros en las funciones deben estar escritos con minúscula separados a un espacio cada uno y en caso de ser compuesto utilizar normativa 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	

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

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

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

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

3.1.2 Diagrama de componentes

Un diagrama de componentes muestra dependencias entre los componentes, que no son más que una unidad física de implementación con interfaces bien definidas pensada para ser utilizada como parte reemplazable de un sistema. Puede mostrar un sistema configurado, con la selección de componentes usados para construirlo o un conjunto de componentes disponibles (una biblioteca de componentes) con sus dependencias.(26)
A continuación se muestra el diagrama de componentes de la solución.

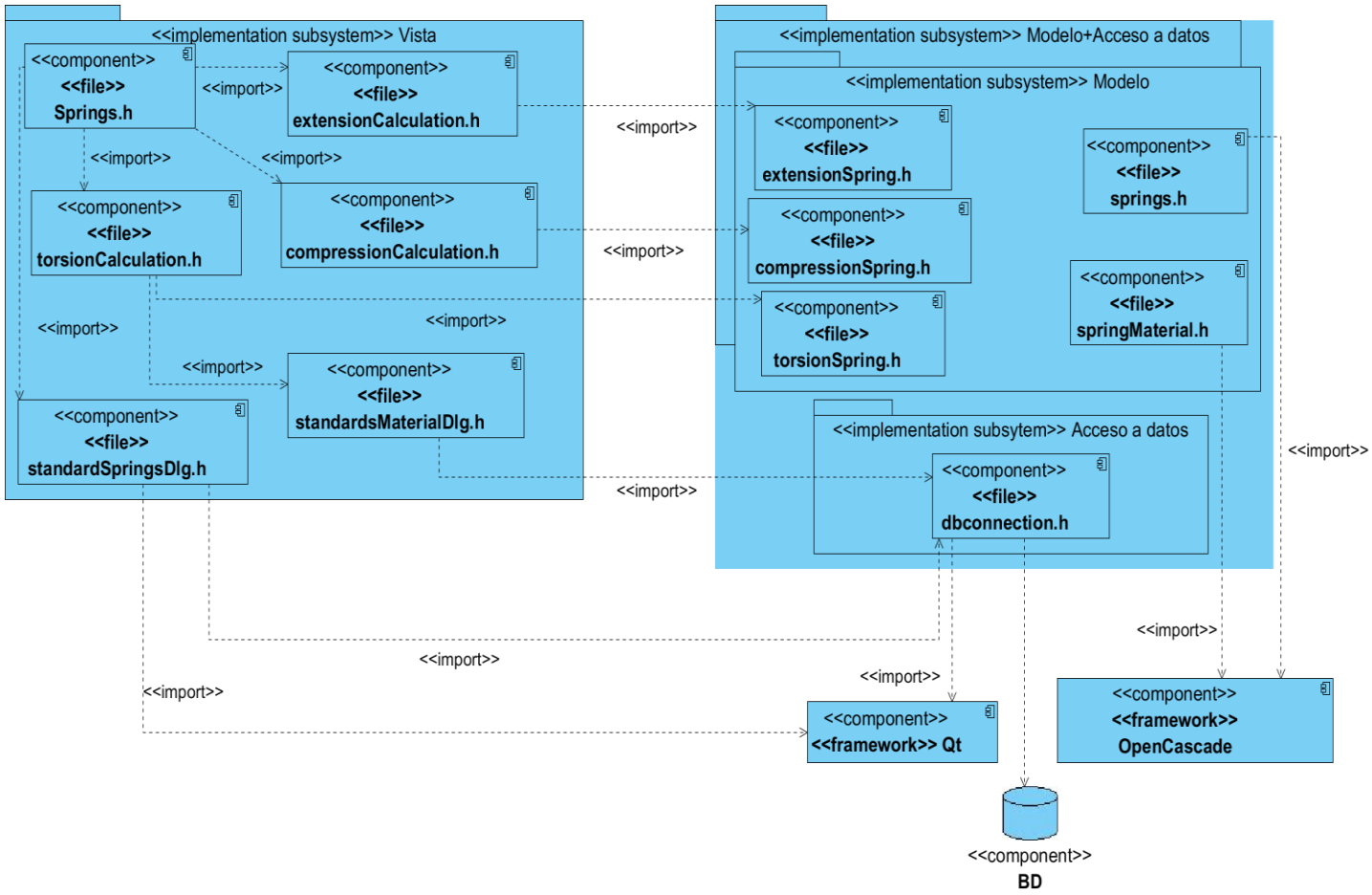


Fig. 11 Diagrama de componentes

3.2 Pruebas

El proceso de pruebas de software tiene dos objetivos distintivos(48):

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

3.2.1 Niveles de prueba

Para aplicarle pruebas a un sistema se deben tener en cuenta una serie de objetivos en diferentes escenarios y niveles de trabajo, debido a que las pruebas son agrupadas por niveles que se encuentran en distintas etapas del proceso de desarrollo. Los niveles de prueba son(54):

- **Prueba Unitaria o de Unidad:** se centra en el proceso de verificación de la menor unidad del diseño del software: el componente de software o módulo. Se emplea para detectar errores debidos a cálculos incorrectos, comparaciones incorrectas o flujos de control inapropiados. Las pruebas del camino básico y de bucles son técnicas muy efectivas para descubrir una gran cantidad de errores en los caminos.
- **Prueba de Integración:** es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es tomar los módulos probados mediante la prueba unitaria y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.
- **Pruebas de Sistema:** está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas. Algunas de estas son: pruebas de recuperación, seguridad, resistencia, entre otras.
- **Pruebas de Aceptación:** es la realización de una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. Un plan de prueba traza la clase de pruebas que se han de llevar a cabo, y un procedimiento de prueba define los casos de prueba específicos en un intento por descubrir errores de acuerdo con los requisitos.

Para validar el correcto funcionamiento de la herramienta se le realizarán pruebas Unitarias, de Sistema y de Aceptación.

3.2.2 Métodos de prueba

El principal objetivo del diseño de casos de prueba es obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software. Para llevar a cabo este objetivo, se emplearán los dos métodos de prueba(54):

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

3.2.3 Diseño de Casos de Prueba

Los Casos de Pruebas han sido realizados sobre la base de las Historias de Usuarios y tienen como objetivo fundamental encontrar la mayor cantidad posible de deficiencias existentes en las funcionalidades implementadas.

A continuación se presenta el Caso de Prueba de la Historia de usuario Seleccionar un material normalizado por una norma determinada, para un mayor estudio de estos ver el Anexo B.

Tabla 7 Caso de Prueba de la Historia de Usuario "Seleccionar un material normalizado"

Descripción general			
Permitir la selección de un material normalizado por una norma determinada.			
SC 1 Seleccionar un material normalizado			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción de Seleccionar un material	Selecciona el campo de selección Standard de la sección del material	Brinda la posibilidad de seleccionar un material normalizado, a partir de los siguientes datos del	Accelerators/Springs/Compression/Calculate/ Standard o Accelerators/Springs/Extension/Calculate/Standard o

normalizado por una norma determinada	y se muestra una ventana para seleccionar el material normalizado, selecciona el nombre de la norma y se muestra la tabla con los datos de los materiales de dicha norma, da doble clic sobre la fila del material deseado y se cierra la venta y actualizan los datos de la vista anterior.	material: coeficiente de relación de esfuerzo tensor con esfuerzo torsor, esfuerzo tensor máximo, módulo de elasticidad a cortante, módulo de torsión, densidad, rango de la fuerza tensora y el nombre del material. Permite además: - Cancelar la operación en cualquier momento.	Accelerators/Springs/Torsion/Calculate/Standard
EC 1.2 Opción de cancelar.	Selecciona la opción de Cancelar.	*Elimina los datos creados.* Cierra la ventana.	Accelerators/Springs/Compression/Calculate/Standard/Cancel o Accelerators/Springs/Extension/Calculate/Standard/Cancel o Accelerators/Springs/Torsion/Calculate/Standard/Cancel

3.2.4 Pruebas Unitarias

Para la realización de las pruebas unitarias se empleó **Qt Test**, el cual es un framework para pruebas de este tipo a aplicaciones y librerías. "**Qt Test** provee todas las funcionalidades comunes de los *framework* de pruebas unitarias, así como extensiones para probar interfaces gráficas de usuario"(55). A continuación se muestra una imagen de las pruebas unitarias realizadas durante el proceso de implementación, en la que se evidencia la aceptación de todas las verificaciones:

```
********* Start testing of TestUnitTest *********
Config: Using QTest library 4.8.6, Qt 4.8.6
PASS   : TestUnitTest::initTestCase()
PASS   : TestUnitTest::testExtensionData()
PASS   : TestUnitTest::testCompressionData()
PASS   : TestUnitTest::testTorsionData()
PASS   : TestUnitTest::testTypeCompression()
PASS   : TestUnitTest::testTypeExtension()
PASS   : TestUnitTest::testTypeTorsion()
PASS   : TestUnitTest::testCompressionFatigue()
PASS   : TestUnitTest::testExtensionFatigue()
PASS   : TestUnitTest::testTorsionFatigue()
PASS   : TestUnitTest::testCorrectionTypeChangeCompression()
PASS   : TestUnitTest::testCorrectionTypeChangeExtension()
PASS   : TestUnitTest::testCorrectionTypeChangeTorsion()
PASS   : TestUnitTest::cleanupTestCase()
Totals: 14 passed, 0 failed, 0 skipped
********* Finished testing of TestUnitTest *********
```

Fig. 12 Figura de resultado de prueba unitaria realizada con Qt Test.

3.2.5 Resultados de las pruebas

La realización de pruebas funcionales permitió identificar una serie de no conformidades:

No. NC	Requisito Funcional	Descripción	Complejidad	Estado
1	RF 10	No se actualizan los datos al cambiar a un material no normalizado.	BAJA	Resuelta
2	RF 2	No se actualizan los campos D1 y D2 si primero el usuario selecciona el tipo de extremo, Double Twisted Full Loop on Side o Full Loop on Side, y luego se introducen los datos del resorte.	BAJA	Resuelta

3	RF 10	No se puede modificar el Allowable Torsional Stress si el material no es estándar.	BAJA	Resuelta
4	RF 1,2,3,7,8,9	No se muestran las unidades de medidas.	BAJA	Resuelta
5	RF 1,2,3	No se muestran los parámetros que poseen error de color rojo.	BAJA	Resuelta
6	RF 2	Si hay errores en las longitudes D1 o D2, si modifico los valores a valores correctos, siguen marcándose en color rojo.	BAJA	Resuelta
7	RF 13	No se señalan los datos incorrectos que se expresan en la sección de los errores de diseño.	BAJA	Resuelta

En la siguiente gráfica se muestran los datos correspondientes a cada iteración de prueba por las que transitó el componente:

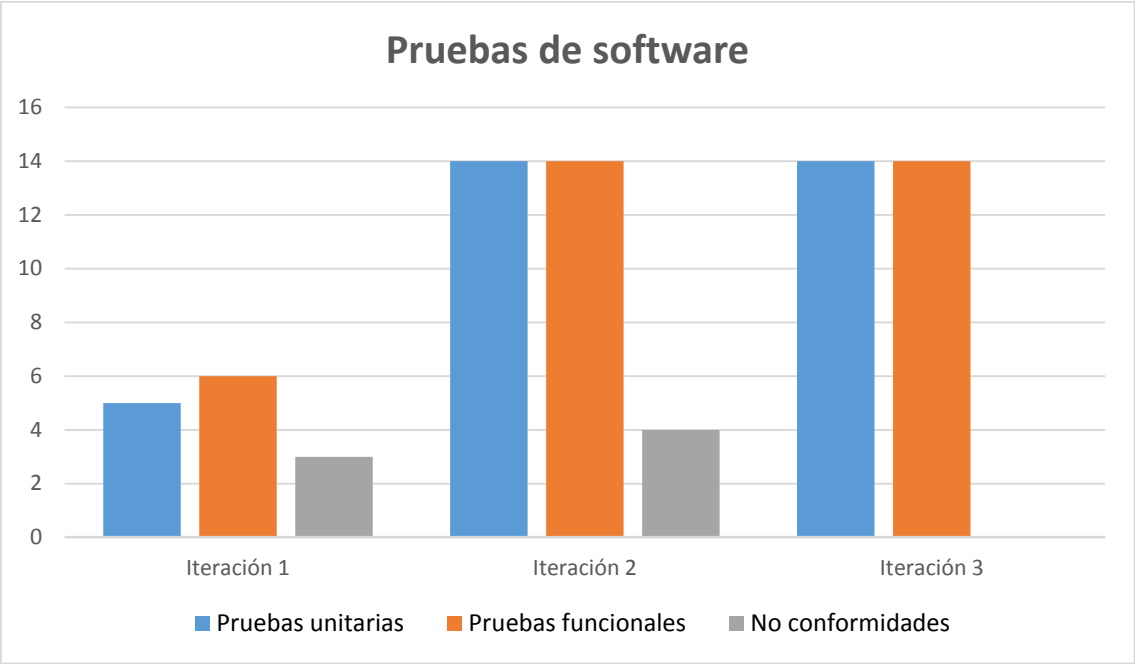


Fig. 73 Gráfico de iteraciones de las pruebas realizadas.

3.3 Conclusiones del capítulo

En el presente capítulo se definió el estilo de codificación a emplear y se generó el diagrama de componente, describiendo con este último los elementos físicos que existen en el sistema y sus relaciones. También, con el fin de validar las funcionalidades desarrolladas se realizaron pruebas al software, las cuales detectaron 7 no conformidades que fueron corregidas, garantizando una mejor calidad de la solución.

Conclusiones generales

Durante el transcurso de la investigación y el proceso de desarrollo se arribaron a las siguientes conclusiones:

- De acuerdo a la información consultada, no existe precedente de un módulo similar ni en el país ni en aplicaciones de código abierto FreeCAD y LibreCAD.
- Se obtuvo un componente que acelera el proceso tradicional de diseño para el caso de los resortes helicoidales, con lo que quedó cumplido el objetivo planteado.
- La solución contiene tres módulos de resortes existentes en los sistemas propietarios.
- El componente se puede ajustar a cualquier aplicación de código abierto que emplee la tecnología Open Cascade.

Recomendaciones

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

- Optimización temporal del modelado de los resortes helicoidales de extensión.
- Investigación de otras alternativas para el cálculo de la fatiga.

Referencias bibliográficas

1. J.E. SHIGLE. *Diseño en Ingeniería mecánica*. 2da. Ediciones Mc Graw Hill, 1979.

2. SANDY GARCÍA SANTOS. *Componente para la modelación de engranajes cilíndricos de dientes rectos*. 2015.

3. JOSÉ MARÍA BALDASANO, SANTIAGO GASSÓ and FERNANDO G. COLINA. DISEÑO ASISTIDO POR ORDENADOR (CAD). EVOLUCION Y PERSPECTIVAS DE FUTURO EN LOS PROYECTOS DE INGENIERIA. [En línea]. [Consultado el: 19 de Enero del 2016]. Disponible en: http://www.unizar.es/aeipro/finder/INGENIERIA%20DE%20PRODUCTOS/BF04.htm

4. Inventor 2014 Help: Design Accelerator. [En línea]. 2014. [Consultado el: 24 de Mayo del 2016]. Disponible en: http://help.autodesk.com/view/INVNTOR/2014/ENU/?guid=GUID-24104648-BE41-49C7-9DD0-1AF2BAFC102E

5. CARLOS GEREZ. *Resortes Mecánicos helicoidales* [En línea].2014. [Consultado el: 19 de Enero del 2016]. Disponible en: http://campus.fi.uba.ar/pluginfile.php/124298/mod_resource/content/0/Resortes%20mec%C3%A1nicos%20helicoidales%20-%20Carlos%20Gerez%2010-14.pdf

6. ENRIQUE MARTÍNEZ LÓPEZ. *Cálculo de resortes helicoidales de compresión*. 2013.

7. WEISSTEIN, Eric W. Helix. [En línea].[Sin fecha]. [Consultado el: 1 de Febrero del 2016]. Disponible en: http://mathworld.wolfram.com/Helix.html

8. EULALIA IZARD ANAYA. *Resortes. Ampliación de cálculo de elementos de máquinas.* [En línea]. [Sin fecha]. [Consultado el: 19 de Enero del 2016]. Disponible en: https://www.academia.edu/8585046/resortes_mec%C3%A1nicos

9. ROBERT STONE. *Fatigue Life Estimates Using Goodman Diagrams* [En línea]. [Sin fecha]. [Consultado el: 19 de Enero del 2016]. Disponible en: http://www.mw-ind.com/pdfs/GoodmanFatigueLifeEstimates.pdf

10. RICHARD G. BUDYNAS and J. KEITH NISBETT. *Mechanical Engineering Design*. 9na Edición. McGraw-Hill, 2011.

11. LIBARDO VANEGAS USECHE. CARGAS VARIABLES - TEORÍA DE FATIGA. *Libardo V. Vanegas Useche. Facultad de Ingeniería Mecánica.* [En línea]. [Sin fecha]. [Consultado el: 28 de Abril del 2016].Disponible en: http://blog.utp.edu.co/lvanegas/

12. A brief introduction to standards. [En línea]. 2016. [Consultado el: 28 de Abril del 2016]. Disponible en: http://www.din.de/en/about-standards/a-brief-introduction-to-standards

13. SEMCO :: Autodesk Inventor | Curso. [En línea]. [Sin fecha]. [Consultado el: 26 de Enero del 2016]. Disponible en: http://www.semco.com.pe/web/curso_autodesk_inventor.jsp

14. Solid Edge Synchronous Technology: Siemens PLM Software. [En línea]. 2016. [Consultado el: 24 de Mayo del 2016]. Disponible en: https://www.plm.automation.siemens.com/en_us/products/solid-edge/design/synchronous-technology.shtml

15. Solid Edge: Siemens PLM Software. [En línea]. 2016. [Consultado el: 26 de Enero del 2016]. Disponible en: http://www.plm.automation.siemens.com/es_sa/products/solid-edge/

16. CATIA. [En línea]. [Sin fecha]. [Consultado el: 26 January 2016]. Disponible en: http://www.3ds.com/es/productos-y-servicios/catiaTabla 8 No conformidades detectadas con las pruebas de caja negra./

17. Productos de visualización de SOLIDWORKS. [En línea]. [Sin fecha]. [Consultado el: 28 de Enero del 2016]. Disponible en: http://www.solidworks.es/sw/products/visualization/solidworks-visualization-overview.htm

18. Simulación-SolidWorks. [En línea]. 2016. [Consultado el: 28 de Enero del 2016]. Disponible en: http://www.solidworks.es/sw/products/simulation/packages.htm.

19. Gestión de datos-SolidWorks. [En línea]. 2016. [Consultado el: 28 de Enero del 2016]. Disponible en: http://www.solidworks.es/sw/products/product-data-management/packages.htm

20. Diseño Eléctrico-SolidWorks. [En línea]. 2016. [Consultado el: 28 de Enero 2016]. Disponible en: http://www.solidworks.es/sw/products/electrical-design/packages.htm

21. The GNU Lesser General Public License, version 3.0 (LGPL-3.0) | Open Source Initiative. [En línea]. [Sin fecha]. [Consultado el: 24 de Mayo del 2016]. Disponible en: https://opensource.org/licenses/LGPL-3.0

22. About FreeCAD - FreeCAD Documentation. [En línea]. [Sin fecha]. [Consultado el: 26 de Enero del 2016]. Disponible en: http://www.freecadweb.org/wiki/index.php?title=About_FreeCAD

23. MITCalc - Mechanical, Industrial and Technical Calculations. [En línea]. [Sin fecha]. [Consultado el: 26 de Enero del 2016]. Disponible en: http://www.mitcalc.com/

24. eFunda: About Us. [En línea]. 2016. [Consultado el: 26 de Enero del 2016]. Disponible en: http://www.efunda.com/about/about.cfm

25. LABORATORIO NACIONAL DE CALIDAD DEL SOFTWARE. *Ingeniería de software: metodologías y ciclos de vida.* 2009.

26. IVAR JACOBSON, JAMES RUMBAUGH and GRADY BOOCH. *El Lenguaje Unificado de Modelado. Manual de Referencia.* 2000.

27. JAMES RUMBAUGH, IVAR JACOBSON and GRADY BOOCH. *El Proceso Unificado de Desarrollo de Software*. 2000.

28. ROBERTH G. FIGUEROA, CAMILO J. SOLÍS and ARMANDO A. CABRERA. *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* [Sin fecha]

29. LETELIER, Patricio and PENADÉS, Mª Carmen. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *www.cyta.com.ar/ta0502/v5n2a1.htm* [En línea]. 15 de Abril del 2006. [Consultado el: 1 de Marzo del 2016]. Disponible en: http://www.cyta.com.ar/ta0502/b_v5n2a1.htm.

30. CHARLES EDEKI. Agile Unified Process. [En línea]. Septiembre del 2013. [Consultado el: 1 de Marzo del 2016]. Disponible en: http://www.ijcsma.com/publications/september2013/V1I304.pdf

31. TAMARA RODRÍGUEZ SÁNCHEZ. *Metodología de desarrollo para la actividad productiva de la UCI*. 2015.

32. ROGER S. PRESSMAN. *Ingeniería de Software. Un enfoque práctico.* 5ta Edición. 2003.

33. Open CASCADE Technology: Overview. [En línea]. 2016. [Consultado el: 2 de Enero del 2016]. Disponible en: http://dev.opencascade.org/doc/overview/html/index.html#OCCT_OVW_SECTION_1

34. BJARNE STROUSTRUP. *The C++ Programming Lenguage*. 2013.

35. About Qt - Qt Wiki. [En línea]. [Sin fecha]. [Consultado el: 19 de Mayo del 2016]. Disponible en: https://wiki.qt.io/About_Qt

36. GNU Operating System. [En línea]. 2007. [Consultado el: 24 de Mayo del 2016]. Disponible en: http://www.gnu.org/licenses/gpl.html

37. OpenGL Overview. [En línea]. [Sin fecha] [Consultado el: 24 de Mayo del 2016]. Disponible en: https://www.opengl.org/about/#1

38. Extensible Markup Language (XML). [En línea]. [Consultado el: 24 de Mayo del 2016]. Available from: https://www.w3.org/XML/

39. JSON. [En línea]. [Sin fecha]. [Consultado el: 24 de Mayo del 2016]. Disponible en: http://www.json.org/

40. CHARLES W. BACHMAN. The Programmer as Navigator. [En línea]. Noviembre de 1973. Vol. 16. [Consultado el: 24 de Mayo del 2016]. Disponible en: http://delivery.acm.org/10.1145/370000/362534/a1973-bachman.pdf?ip=10.35.30.73&id=362534&acc=OPEN&key=223837E73163AEDA.7CA22F5EB1578DD1.4D4702B0C3E38B35.6D218144511F3437&CFID=796058433&CFTOKEN=97143954&__acm__=1465220368_1d908bb9ef0f67bd58204fdbf85351e0

41. Sobre PostgreSQL | www.postgresql.org.es. [En línea]. [Sin fecha] [Consultado el: 27 de Abril del 2016]. Disponible en: http://www.postgresql.org.es/sobre_postgresql

42. LUIS ALBERTO CASILLAS SANTILLÁN, MARC GIBERT GINESTÁ and ÓSCAR PÉREZ MORA. *Base de datos en MySQL* [En línea]. [Sin fecha]. [Consultado el: 24 de Mayo del 2016]. Available from: http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02151.pdf

43. About SQLite. [En línea]. [Sin fecha]. [Consultado el: 27 de Abril del 2016]. Disponible en: https://www.sqlite.org/about.html

44. HANS-ERIK ERIKSSON and MAGNUS PENKER. *Business Modeling with UML: Business Patterns at Work.* 2000. John Wiley & Sons.

45. Definition of Shear Force and Bending Moment - Civil Engineering Terms. [En línea]. [Sin fecha]. [Consultado el: 23 de Mayo del 2016]. Disponible en: http://www.civilengineeringterms.com/mechanics-of-solids-1/definition-of-shear-force-and-bending-moment/

46. SEARS ZEMANKS and YOUNG FREEDMAN. *Física universitaria*. 9na Edición. Félix Varela, 2008.

47. ROBERT RESNICK, DAVID HALLIDAY and KENNETH S. KRANE. *Física*. 4ta Edición. 2001.

48. IAN SOMMERVILLE. *Ingeniería de Software*. 8va Edición. 2006.

49. R. JEFFRIES, A. ANDERSON and C. HENDRICKSON. *Extreme Programming Installed*. 2001. Addison-Wesley.

50. ROGER S. PRESSMAN. *Ingeniería del software. Un enfoque práctico.* 6ta Edición. 2005.

51. DOLORS COSTAL COSTA. *Introducción al diseño de bases de datos.* [En línea]. [Sin fecha] [Consultado el: 23 de Mayo del 2016]. Disponible en: http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02150.pdf

52. JOSEPH E. SHIGLEY and CHARLES R. MISCHKE. *Standard Handbook of Machine Design*. 2da Edición. McGraw-Hill, 1996.

53. YUKIO ISHII and MASAO MIZUNO. *Stress Correction Factor for Helical Springs*. Universidad de Keio, 1971.

54. ROBERTO RUIZ TENORIO. *Las Pruebas de Software y su Importancia en las Organizaciones*. 2010.

55. Qt Test Overview | Qt Test 5.6. [En línea]. [Sin fecha]. [Consultado el: 16 de Mayo del 2016]. Disponible en: http://doc.qt.io/qt-5/qtest-overview.html

56. CURTIS WAGUESPACK, SEAN DOTSON, P.E., BILL BOGAN, ANDREW FAIX, SETH HINDMAN, LOREN JAHRAUS, P.E., DENNIS JEFFREY, SHEKAR SUBRAHMANYAM y BOB VAN DER DONCK. *Mastering Autodesk Inventor 2009 and Autodesk Inventor LT 2009*. Wiley Publishing, Inc., 2008.

57. How to create spring. [En línea]. [Sin fecha]. [Consultado el: 17 de Mayo del 2016]. Disponible en: http://www.solidworkstutorials.com/solidworks-spring/

58. MITCalc - Compression Springs Download. *softpedia* [En línea]. [Sin fecha]. [Consultado el: 17 de Mayo del 2016]. Disponible en: http://www.softpedia.com/get/Science-CAD/MITCalc-Compression-Springs.shtml

59. Springulator® Spring Calculator | Calculator Springs | Newcomb Spring. [En línea]. [Sin fecha]. [Consultado el: 17 de Mayo del 2016]. Disponible en: http://www.springulator.com/

60. Calculator for Designing Compression Springs. [En línea]. 2016. [Consultado el: 17 de Mayo del 2016]. Disponible en: http://www.efunda.com/designstandards/springs/calc_comp_designer.cfm

61. *Compression springs. Dimensions according to DIN 2098.* [En línea]. [Sin fecha]. [Consultado el: 17 de Mayo del 2016]. Disponible en: http://www.lesjoforsab.com/standard-springs/35-42_en_id962.pdf