Universidad de las Ciencias Informáticas
Facultad 4

Módulo para automatizar el proceso de diseño de mecanismos de levas

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

Autor: Reydel Baños Acosta.

Tutores: Dr.C Augusto César Rodríguez Medina.

Ing. Gustavo García González.

La Habana, 2018

"Aprendí que el coraje no es la ausencia de miedo, sino el triunfo sobre él. El hombre valiente no es aquel que no siente miedo sino el que conquista ese miedo."

Nelson Mandela

Dedicatoria

A mis abuelos Marta y Guillermo y a mi abuela Enma por ser los mejores, por acompañarme en toda mi vida estudiantil, además por acompañarme espiritualmente.

Agradecimientos

A mis padres y mi familia por apoyarme a lo largo de los 5 años de la carrera A mi tutor Augusto y mi Cotutor Gustavo un Agradecimiento muy especial, A todos mis amigos y compañeros de Aula y a todos los profesores que aportaron a mi formación profesional

Declaración de autoría

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

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

Reydel Baños Acosta.
Autor

Dr.C Augusto César Rodríguez Medina.
Tutor

Ing. Gustavo García González.
Tutor

Resumen

En el presente documento se exponen los resultados de un proceso de investigación, orientado al desarrollo de un módulo informático para acelerar el diseño de perfiles de levas, empleando tecnologías de código abierto disponibles como Qt y *Open CASCADE Technology*. La formulación empleada para el cálculo de las diferentes variantes de levas fue definida a partir de las normas vigentes y la literatura especializada. La importancia del trabajo reside en que ofrece una alternativa al empleo de sistemas comerciales a los que Cuba no puede acceder, debido a las restricciones del bloqueo impuesto por el Gobierno de Estados Unidos y a los elevados precios que tienen en el mercado. El resultado obtenido posee funcionalidades para el modelado de tres tipos de levas: de disco, lineales y cilíndricas, permitiendo reducir el tiempo de diseño en relación a los procedimientos tradicionales. También contribuye a la soberanía tecnológica en el área del diseño asistido por computadora

Palabras clave: Acelerador de diseño, diseño asistido por computadora, mecanismos de levas.

Índice general

Índice de figuras

Índice de tablas

Introducción

Una leva es un elemento mecánico que sirve para impulsar a otro llamado seguidor que tendrá un movimiento determinado por contacto directo. Los mecanismos de leva y seguidor son sencillos y poco costosos, tienen pocas piezas móviles y ocupan espacios muy reducidos (Rothbart, 2004). Las levas desempeñan un papel importante dentro de la maquinaria moderna y se han empleado extensamente dentro de motores de combustión interna, máquinas herramientas, computadores mecánicos y otras muchas aplicaciones. Una de las tendencias en el desarrollo de la técnica moderna es el intento de reducir los tiempos de cualquier tipo de proceso y particularmente los de diseños (Norton, 2009). Se puede apreciar que en ocasiones, lo anterior se alcanza mediante diferentes formas de automatización. En el caso de los sistemas informáticos para el diseño se han desarrollado módulos que aceleran el proceso de cálculo y modelado de elementos de máquinas, incluyendo los de leva.

La situación de la industria cubana en relación con las tecnologías para el diseño asistido por computadora, se caracteriza por la negación del acceso a estas , debido a las restricciones del bloqueo impuesto por el gobierno de los Estados Unidos contra Cuba (Montes de Oca Montano, 2015); en este caso, la consecuencia directa es la imposibilidad de automatizar los procesos de diseño en ingeniería, lo que genera afectaciones a la productividad y el incremento del costo de los productos que se proyectan.

Como país subdesarrollado, Cuba podría tener limitaciones para adquirir licencias de sistemas informáticos dedicados al diseño aunque fuera levantado el bloqueo, debido a los elevados precios de estas, que pueden ascender a 4000 USD por estación de trabajo (ver tabla) 1 . Invertir en el área de las tecnologías para el diseño no solo es importante para incrementar la productividad en ese proceso, sino también para implementar el trabajo colaborativo de ingeniería por internet, que ya es una práctica generalizada en países desarrollados, y comercializar proyectos.Una alternativa para contribuir a la solución del problema planteado, puede ser el desarrollo de módulos para automatizar los cálculos repetitivos y acelerar el diseño de mecanismos de leva.

La factibilidad del empeño mencionado en el párrafo precedente está demostrada por antecedentes concretos, como el desarrollo de la solución ASIXMEC(ASIXMEC, s.f.) y de módulos para acelerar el diseño de componentes mecánicos .

Como recurso disponible, que aumenta la posibilidad de éxito en el proceso de desarrollo, está la existencia de aplicaciones para el diseño y la ingeniería asistidos por computadora como FreeCAD y Salome-Meca, ambas de código abierto y basadas en la tecnología Open CASCADE; al estar disponible el código fuente es factible estudiar funcionalidades de modelado importantes y características de su arquitectura para aplicarlas

a la solución proyectada.

Tabla 1. Tabla de costo de las licencias de los *software* comerciales (Asidek, 2018a). (Asidek, 2018b), (Araworks, 2018), (Simiens PLM, 2018)

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

Teniendo en cuenta lo antes expuesto se formuló como **problema a resolver** la baja productividad en el proceso de diseño de mecanismos de leva, por no disponerse en el país de tecnologías apropiadas para automatizarlo.

El **objeto de estudio** está asociado a la línea gráficos por computadora para aplicaciones industriales, y el **campo de acción** está delimitado en los algoritmos para el cálculo y modelado de mecanismos de leva.

Para la solución del problema se planteó como **objetivo general**: Desarrollar e implementar un módulo para acelerar el diseño de mecanismos de leva con tecnologías de código abierto.

Los **objetivos específicos** son:

1. Establecer el marco referencial relacionado con la línea de gráficos por computadora para los algoritmos de cálculo y modelado de mecanismos de leva con tecnologías de código abierto.
2. Diseñar los algoritmos para implementar las funcionalidades de modelado de mecanismos leva.
3. Implementar un módulo para acelerar el proceso de diseño de los distintos tipos de mecanismos de levas.

Para dar cumplimiento a estos objetivos específicos se proponen las siguientes **tareas**, que se agrupan en las siguientes áreas:

Marco Teórico.

- Investigación preliminar destinada a precisar el estado del arte en relación al tema, y el análisis sobre la factibilidad de generar la solución.
- Elaboración del diseño de la investigación.

Análisis

- Asimilar la fundamentación teórica en que se basa el diseño de los mecanismos de levas.

- Estudio de las funcionalidades de los módulos para el diseño de mecanismos de levas existentes en aplicaciones comerciales de diseño.
- Análisis de los códigos fuente de las aplicaciones FreeCad y Salome-Meca con la finalidad de identificar y evaluar la reutilización de sus funcionalidades, así mismo, identificar las funcionalidades de la tecnología OpenCascade que serán utilizadas en el proceso de desarrollo.
- Proceso de síntesis (obtención de conclusiones, resultados, definición de las formas de hacer y conformación de la estructura del componente).
- Estudio de los aspectos de la metodología de desarrollo de software (AUP-UCI) que se aplicarán en la investigación.
- Obtención de requisitos a partir de los módulos para el diseño de mecanismos de leva existentes en sistemas propietarios y por las consideraciones de usuarios potenciales.

Diseño

- Definición de la arquitectura del componente, lo que implica además definir los patrones de diseño con los que cumplirá.
- Elaboración de la estructura de clases del componente.
- Diseño de la interfaz gráfica del componente.

Implementación

- Implementación del módulo para acelerar el proceso de diseño de mecanismos de levas.
- Realización de pruebas al componente.

Para resolver y dar cumplimiento a los objetivos y tareas propuestas se emplearon los siguientes **métodos de investigación:**
Métodos teóricos:

- **Análisis y síntesis**: se emplearon durante el estudio de los fundamentos teóricos de la construcción de los mecanismos de leva.

Métodos empíricos:

- **Observación**: se empleó durante la identificación de requisitos y en la etapa de pruebas.
- **Medición**: durante las pruebas de funcionamiento se toman los tiempos de modelado y cálculo para poder estimar de manera objetiva la eficiencia del módulo.

El presente documento constituye el informe técnico de la investigación realizada y está estructurado en tres capítulos; en el primero se exponen aspectos generales de la fundamentación teórica como la caracterización de los sistemas Diseño Asistido por Computadora (CAD, por sus siglas en inglés) y los módulos que poseen, las características de los mecanismos de leva, así como las metodologías y herramientas para el desarrollo. En el segundo capítulo,se define la propuesta de solución que contiene la descripción del

componente con sus requerimientos en cuanto arquitectura, patrones de diseño, requisitos funcionales y no funcionales. En el último capítulo se presentan los estándares de codificación, diagrama de componente, el resultado obtenido y las pruebas efectuadas al mismo.

CAPÍTULO 1

Fundamentos de la investigación, metodología y tecnologías

En el presente capítulo se presenta la terminología y nomenclatura empleada en la teoría de levas, para facilitar el seguimiento del texto, que incluye figuras y gráficas como apoyo a las explicaciones. De igual forma se expone información esencial sobre sistemas similares a la solución proyectada en el presente trabajo y los fundamentos teóricos asociados al diseño de mecanismos de leva; finalmente, se mencionan las metodologías y las herramientas de desarrollo a emplear.

1.1. Mecanismos de levas

Como ya se ha definido, una leva es un elemento mecánico que sirve para impulsar a otro llamado seguidor, para que desarrolle un movimiento especificado por contacto directo; el movimiento, como regla, sigue una ley de desplazamiento determinada (Rothbart, 2004).

1.1.1. Características de los mecanismos levas

Los mecanismos de leva se han construido históricamente de materiales diversos (madera, metal, plástico, etc.), se montan sobre un eje y su característica principal es un contorno especial o en ocasiones una ranura, que guía el movimiento de un dispositivo denominado seguidor (ibíd.), cada componente tiene un grado de libertad en su movimiento y una coordenada generalizada independiente ql y qp para la leva y el seguidor respectivamente, que se ponen en contacto. De esta manera se obtiene un mecanismo gobernado por una ley de dependencia $qp(ql)$ entre las coordenadas generalizadas que describe el movimiento de la leva y del seguidor, esta recibe la denominación de ley de desplazamiento. El movimiento de la leva, usualmente de rotación, se transforma en movimiento de rotación o traslación del seguidor. Los mecanismos de leva-seguidor generalmente son utilizados como generadores de funciones; son sencillos, poco costosos, tienen pocas piezas móviles y ocupan espacios muy reducidos. Las leyes de desplazamiento del seguidor no son difíciles de diseñar y puede lograrse casi cualquier configuración que se desee; los perfiles de leva que proporcionan el movimiento al seguidor según la ley requerida son mecanizados actualmente por máquinas

de control numérico, ver figura 1.1

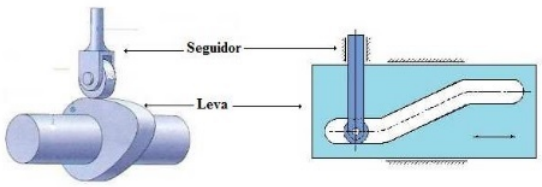


Figura 1.1. Mecanismo de leva-seguidor
(Serrano Muñoz, 2010)

1.1.2. Clasificación de los mecanismos leva

Las levas pueden clasificarse atendiendo diferentes criterios. A continuación se exponen los más comunes en la figura 1.2:

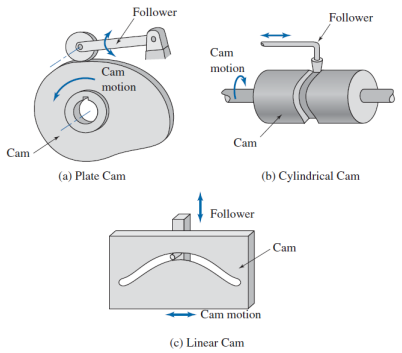


Figura 1.2. Tipos de levas a automatizar

En la figura 1.3 puede observarse una leva plana de rotación con seguidor traslacional de rodillo.

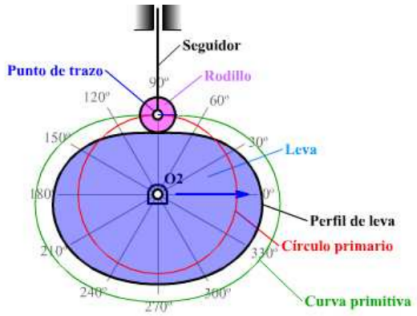


Figura 1.3. Leva plana de rotación con seguidor traslacional de rodillo

En la figura se ilustran los conceptos que a continuación se detallan::

- Punto de trazo: viene dado por el centro del rodillo que hace de seguidor.
- Rr: radio del rodillo del seguidor.
- Circunferencia base: es la más pequeña que puede trazarse tangente a la superficie de la leva.
- Rb: radio de la Circunferencia base.
- Curva de paso o primitiva: es la curva o trayectoria que describe el punto de trazo del seguidor al moverse siguiendo el perfil de la leva.
- Punto de trazo: viene dado por el centro del rodillo o por el extremo del seguidor según sea el caso.
- Rr: radio del rodillo del seguidor.
- Circunferencia primaria: es la más pequeña que puede trazarse tangente a la curva de paso.
- Rp: radio de la circunferencia primaria. Se puede observar que $Rp = Rf + Rb$.

La figura 1.4 representa una leva plana de rotación con seguidor de cara plana con movimiento de traslación; en este caso la leva es más simple, pues desaparecen la curva de paso y el círculo primario.

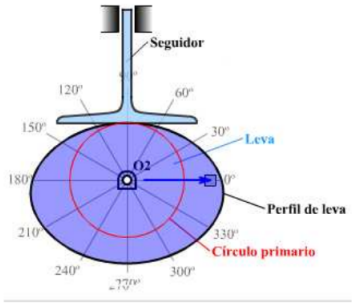


Figura 1.4. Leva plana de rotación con seguidor traslacional de cara plana.

1.1.3. Levas alternativas o traslacionales

En la bibliografía consultada, a este tipo de levas se les da diversos nombres: levas de traslado, de cuña, traslacionales, lineal, etc.; todas estas denominaciones hacen referencia a las levas cuyo contorno o forma realiza un movimiento de traslación y dónde esta forma base quedará determinada por el movimiento específico del seguidor. La denominación de levas alternativas se debe a que realizan un movimiento traslacional de manera alternativa. Estas levas generan, con su movimiento, la traslación del seguidor; debido a esta acción alternativa tiene lugar un viaje de ida y otro de retorno en el cual se sigue el mismo movimiento en el orden inverso, por lo que el perfil de la leva quedará definido con la propia ley de desplazamiento que se defina para el seguidor (Yañez-Valdeza et al., s.f.). En el caso de las levas alternativas, se utilizan los siguientes conceptos:

- Punto de trazo: viene dado por la trayectoria del centro del rodillo en el extremo del seguidor.
- Rr: radio del rodillo del seguidor.
- Curva de paso o primitiva: es la trayectoria que describe el punto de trazo del seguidor al moverse siguiendo el perfil de la leva.

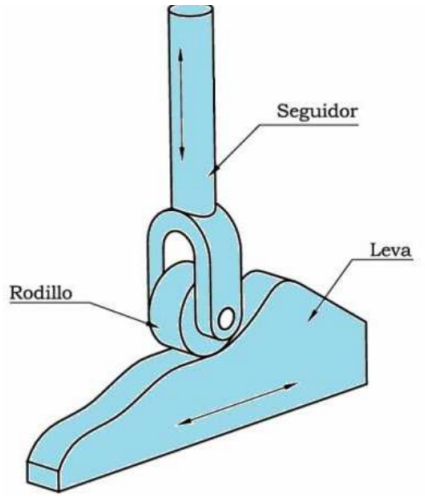


Figura 1.5. Leva alternativa o traslacional.

1.1.4. Secuencia de diseño de un mecanismo leva-seguidor

La propia definición de mecanismo leva-seguidor expuesta al inicio, permite establecer la misión del mismo, que consiste en impulsar al seguidor según una ley de desplazamiento $qp(ql)$ deseada, en función de la coordenada independiente ql de la leva.

La secuencia de diseño de un mecanismo leva-seguidor en lo que se refiere a aspectos geométricos y cinemáticos según (Cardona y Clos Acosta, 2001) es la siguiente: Diseño de la ley de desplazamiento $qp(ql)$. Hay dos tipos de restricciones de movimiento: la primera, conocida como **CEP** (Critical Extreme Position)

establece las condiciones de contorno para el seguidor al comienzo y al final de los tramos del diagrama de desplazamiento; la segunda, conocida como **CPM**. (Critical Path Motion), requiere también el paso por determinadas posiciones intermedias en el presente trabajo se utilizó el primer tipo de restricciones.

Al tratarse de un proceso cargado de cálculos y que suele requerir de iteraciones hasta llegar a una solución aceptable que se adapte a las especificaciones, es usual en la actualidad, el uso de herramientas informáticas que faciliten la tarea y ofrezcan la posibilidad de realizar los cálculos analíticos con gran rapidez; mediante procedimientos manuales el proceso de diseño de mecanismos de leva puede tomar decenas de horas de trabajo a un diseñador según los métodos que emplee, si lo hace por análisis podría ser más sencillo, pero si es por síntesis el proceso es más complicado; asumiendo que requiera 36 horas de trabajo hasta completar la documentación técnica, comparado con una solución computacional, se puede constatar el incremento en la eficiencia del proceso. Una posibilidad viable y efectiva para acelerar el proceso de diseño de mecanismos de leva puede ser el desarrollo de un módulo que contenga las funcionalidades para modelar levas de disco, lineales y cilíndricas.

1.1.5. Consideraciones sobre la función de desplazamiento

El primer paso en el cálculo de una leva es la especificación del movimiento de salida que debe gobernar al seguidor, que como ya se ha definido recibe la denominación de ley o función de desplazamiento; en el diseño de una ley de desplazamiento $qp(ql)$ para el seguidor existen algunas especificaciones obligatorias y otras recomendables. La elección de las especificaciones no impuestas por la funcionalidad es el aspecto al que debe prestársele mayor atención, ya que la ley de desplazamiento $qp(ql)$ resultante determinará además del contorno de la leva, la cinemática del seguidor, y por tanto, la dinámica del mecanismo.

Los mecanismos leva-seguidor más utilizados en la práctica son aquellos en los que la leva tiene movimiento de rotación, ya que este es fácil de obtener; de esta manera la coordenada generalizada independiente ql que caracteriza a la leva puede llamarse, ángulo de giro de la leva; la coordenada generalizada independiente qp que caracteriza al seguidor será s si el seguidor tiene movimiento de traslación y θ si tiene movimiento de rotación, de este modo, la ley de desplazamiento del seguidor será $s(\theta)$ para mecanismos en los que el seguidor se traslada y θ para aquellos en los que rota (Correia et al., 2000).

El movimiento de salida se construye por la unión de diferentes tramos comúnmente empleados en el cálculo de perfiles levas; la longitud de cada tramo estará dada por el ángulo parcial de rotación dentro de una vuelta completa, a este ángulo se le denomina θ; la suma de los ángulos parciales de rotación debe ser exactamente 360°. Una ley de desplazamiento se supone construida cuando se han determinado todos los tramos necesarios para completar una vuelta; la información que debe contener cada uno de dichos tramos es(ibíd.) :

- El tipo de movimiento para el tramo: armónico simple, cicloidal, velocidad constante, entre otros.
- El ángulo parcial de rotación θ sobre el que se construye el tramo.
- La magnitud del movimiento lineal (seguidores traslacionales) o angular(seguidores oscilantes) y su sentido. A este valor se le llamará L.

Al representar la ley de desplazamiento $S(\theta)$ gráficamente en un sistema de coordenadas, colocando la variable independiente en el eje de las abscisas y la dependiente en el eje de las ordenadas, se obtiene el diagrama de desplazamiento. En la figura 1.6 puede verse una función de desplazamiento para un seguidor transnacional formada por tres tramos de 120°: el primero es un ascenso del seguidor de 50 mm, le sigue una posición constante para un intervalo de 120°y finalmente en el tercer tramo regresa a la posición inicial.

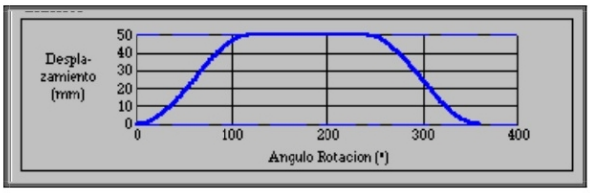


Figura 1.6. Diagrama de desplazamiento

Para poder determinar el tipo de movimiento más adecuado se debe:

- Conocer o identificar las restricciones CPM para el intervalo.
- No valorar aisladamente cada intervalo, sino de forma conjunta con sus adyacentes.
- Conocer bien las características de cada tipo de movimiento, para saber si es o no adecuado dentro de las condiciones del problema.

1.1.6. Relación cinemática leva-seguidor

La velocidad, aceleración y sobreaceleración del seguidor se determinan a partir de la ley de desplazamiento mediante derivaciones sucesivas:

$$S = y(\theta) \tag{1.1.1}$$

$$V = \left(\frac{\partial s}{\partial t} = \frac{\partial y}{\partial \Theta} \cdot \frac{\partial \Theta}{\partial t} = \frac{\partial y}{\partial \Theta}\omega\right) \tag{1.1.2}$$

$$A = \left(\frac{\partial V}{\partial t} = \frac{\partial y^2}{\partial \Theta^2} + \left(\frac{\partial \Theta}{\partial t}\right)^2 + \frac{\partial y}{\partial \Theta} \cdot \frac{\partial^2 \Theta}{\partial t^2} = \frac{\partial^2 y}{\partial \Theta^2}\omega^2 + \frac{\partial y}{\partial \Theta}\alpha\right) \tag{1.1.3}$$

$$J = \left(\frac{\partial A}{\partial t} = \frac{\partial^3 y}{\partial \Theta^3} + \left(\frac{\partial \Theta}{\partial t}\right)^3 + 2 \cdot \frac{\partial^2 y}{\partial \Theta^2} \cdot \frac{\partial \Theta}{\partial t} = \frac{\partial^2 \Theta}{\partial t^2} + \frac{\partial^2 y}{\partial \Theta^2}\frac{\partial \Theta}{\partial t}\frac{\partial \Theta^2}{\partial t^2} + \frac{\partial y}{\partial \Theta}\frac{\partial \Theta^3}{\partial t^3} = \frac{\partial^3 y}{\partial \Theta^3}\omega^3 + 3 \cdot \frac{\partial^2 y}{\partial \Theta^2} \cdot \omega\alpha + \frac{\partial y}{\partial \Theta} \cdot \varphi\right) \tag{1.1.4}$$

S,V,A,J: desplazamiento, velocidad, aceleración y sobreaceleración del seguidor.

1.1.7. Ley fundamental de continuidad

Para que el cálculo de una leva sea aceptable, no se deben producir saltos en el seguidor a causa de un perfil abrupto causado por discontinuidades en la ley de desplazamiento; la continuidad en la función de desplazamiento del seguidor es indispensable y no puede violarse en ningún momento, sin embargo, no es una condición suficiente, y las leyes de desplazamiento han de verificar ciertas condiciones de continuidad en el tiempo, en particular en las uniones entre tramos.(Cardona y Clos Acosta, 2001); es imprescindible que la velocidad del seguidor sea una función continua, lo cual significa que su ley de desplazamiento $S(\theta)$ sea una función al menos de tipo $C1$; las discontinuidades en la velocidad originan aceleraciones infinitas y fuerzas elevadas que podrían, excepto en máquinas muy lentas, conducir a la destrucción del mecanismo o a la pérdida del contacto entre los componentes cuando el cierre del par es por fuerza .

Es muy conveniente, que la aceleración sea una función continua, lo que implica que la función de desplazamiento del seguidor sea una función al menos de tipo $C2$, si la aceleración tiene saltos bruscos, habrían sobreaceleraciones teóricas infinitas que producirían variaciones importantes en las fuerzas que actúan sobre el mecanismo; a causa de la elasticidad y los juegos surgen ruidos y vibraciones que darían lugar a roturas o desajuste entre el movimiento real del seguidor y el requerido (Rothbart, 2004). $C1$ $C2$ son contantes
Son aceptables discontinuidades en la tercera derivada, teniendo en cuenta que esto puede provocar vibraciones que se verán agravadas al aumentar la velocidad de giro de la leva, por lo que en el caso de velocidades angulares elevadas no debe hacerse tal excepción. El movimiento de una leva es cíclico, por lo que la continuidad debe estar asegurada también entre el punto inicial y su unión con el punto final (que son en sí el mismo punto), por lo que cuando la leva completa una vuelta (360°), el desplazamiento y las dos primeras derivadas (como mínimo) deben coincidir en 0.

1.1.8. Valores pico de velocidad y aceleración

En ocasiones, la elección del tipo de movimiento para un tramo pasa por la incertidumbre de seleccionar entre diferentes posibilidades aparentemente válidas; en ese caso es necesario desechar aquellas que produzcan magnitudes pico de velocidad y aceleración superiores a las demás, mayores velocidades implican mayor energía cinética y consumos superiores de potencia, aceleraciones mayores producen cargas dinámicas que afectan a todo el sistema y que se deben tratar de minimizar para alargar el ciclo de vida de la leva y su seguidor (García, 2008).

1.2. Funciones matemáticas que definen la ley de desplazamiento

En este apartado se explican las características que definen a cada una de las funciones empleadas en el cálculo de levas, para construir la ley de desplazamiento del seguidor, y a partir de ella las leyes de velocidad, aceleración y sobreaceleración. Cada ley de desplazamiento es creada por la unión de diferentes tramos de manera que se cumplan las condiciones de contorno requeridas. Cada tramo opera dentro de un intervalo angular englobado en la rotación total de la leva. Al ser la función de desplazamiento un conjunto

de funciones, lo que se hace es tratar cada intervalo independientemente dentro de sus límites de rotación usando para ello una variable adimensional que será fracción de dos ángulos: el numerador θ se corresponde con el ángulo parcial recorrido dentro de su intervalo; el denominador (ya explicado) es la longitud total del tramo angular de rotación, así,θ/B varía de 0 a 1, siendo θ el punto inicial y 1 el punto final del tramo. La curva de desplazamiento será parecida para casi todos los movimientos, sin embargo, muestra diferencias en sus derivadas asociadas. Las condiciones de contorno y los valores pico de las mismas no precisan ser iguales, ambos serán los criterios principales para decidirse por un tipo de movimiento o por otro, buscando el conjunto de tramos que cumpliendo las restricciones del problema aseguren la continuidad, como mínimo en velocidad y aceleración, después seleccionando entre las varias alternativas posibles para un posible tramo. Otro cálculo es necesario para precisar la forma, magnitudes geométricas y movimiento del seguidor y también las de la leva, lo que permitirá determinar el perfil de la leva para generar el movimiento de salida del seguidor. Con ayuda de ciertos parámetros se determinará si el perfil de la leva satisface las exigencias de diseño, si no es así se recalcula modificando cualquier dato de partida que pueda alterar la solución anterior(Haug, 1989).

Las curvas comúnmente empleadas en el cálculo cinemático de levas son las siguientes(Serrano Muñoz, 2010):

- Armónico Simple (sólo para ascenso y descenso consecutivos, ambos con B=180°o polinómicas genéricas de 5^{to} ó 7^{mo} grado).
- Cicloidal.
- Polinómico de 5^{to} grado (caso particular: polinómico 3-4-5).
- Polinómico de 7^{mo} grado (caso particular: polinómico 4-5-6-7).
- Velocidad constante.
- Movimientos de medio período (semiarmónicos y semicicloidales).
- Armónico doble (sólo para ascenso y descenso consecutivos, ambos con igual o con polinómicos genéricos de 5^{to} o 7^{mo} grado).
- Movimientos compuestos (Aceleración de onda sinusoidal modificada y aceleración de onda Trapezoidal modificada).
- Detenimiento.

1.2.1. Armónico Simple

En la figura 1.7 se presenta el movimiento armónico simple, la construcción gráfica utiliza una semicircunferencia con diámetro igual a la elevación L; se divide el eje de las abscisas en el número de partes en que se divide la semicircunferencia.

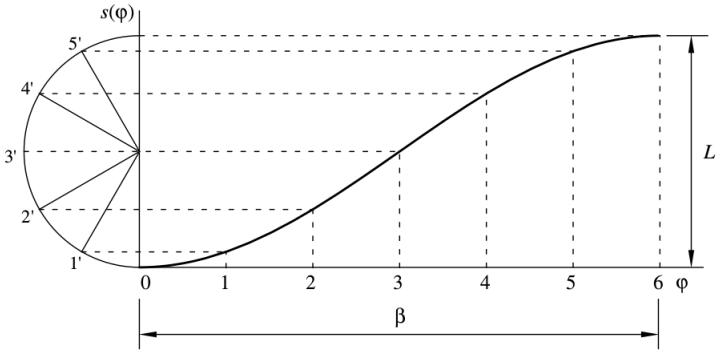


Figura 1.7. Diagrama de un movimiento armónico simple
(Rothbart, 2004)

Características de la curva:

La ventaja de esta curva es la uniformidad de la velocidad y la aceleración durante el recorrido, sin embargo, los cambios instantáneos de la aceleración al principio y al final del movimiento suelen provocar vibración, ruido y desgaste. En este caso las ecuaciones que describen el movimiento son (Rothbart, 2004):

$$\textbf{Desplazamiento} : fy(z) = 0.5 * (1 - cos\pi * z)) \tag{1.2.1}$$

$$\textbf{Velocidad} : fv(z) = 0.5\pi sin(\pi * z) \tag{1.2.2}$$

$$\textbf{Aceleración} : fa(z) = 0.5\pi^2 cos(\pi * z) \tag{1.2.3}$$

$$\textbf{Impulso} : fj(z) = -0.5\pi^3 sin(\pi * z) \tag{1.2.4}$$

1.2.2. Cicloidal

En la figura 1.8 se presenta la construcción del movimiento cicloidal, esta ley debe su nombre a la cicloide que es la trayectoria de un punto P de un círculo de radio $r = L/2\pi$, siendo L la elevación, cuando gira sin deslizar sobre la ordenada del diagrama de desplazamiento (ibíd.).

Ecuaciones del movimiento:

$$\textbf{desplazamiento} : fy(z) = z - 0.5/\pi \cdot sin(2\pi \cdot z) \tag{1.2.5}$$

$$\textbf{Velocidad} : fv(z) = 1 - cos(2\pi * z)) \tag{1.2.6}$$

$$\textbf{Aceleración} : fa(z) = 2 \cdot sin(2 \cdot \pi \cdot z)) \tag{1.2.7}$$

$$\textbf{Impulso} : fj(z) = 4 * \pi^2 * cos(2 * \pi * z)) \tag{1.2.8}$$

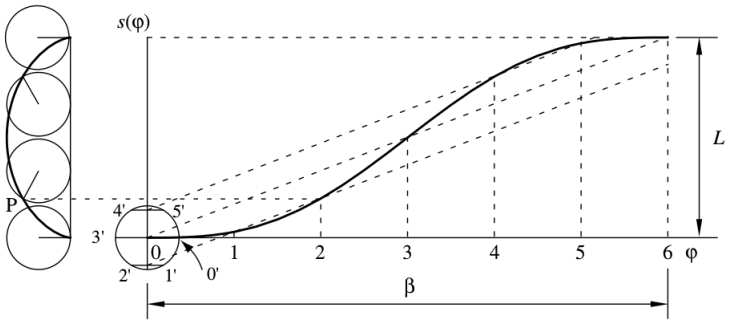


Figura 1.8. Diagrama de un movimiento cicloidal

1.2.3. Lineal

En la figura 1.9 se presenta la construcción del movimiento Lineal Movimiento simple con un enorme impacto al principio y al final del movimiento. No se usa casi nunca, salvo en dispositivos muy rudimentarios (Rothbart, 2004) .

$$\textbf{Desplazamiento} : fy(z) = Z \tag{1.2.9}$$

$$\textbf{Velocidad} : fv(z) = 1 \tag{1.2.10}$$

$$\textbf{Aceleración} : fa(z) = 0 \tag{1.2.11}$$

$$\textbf{Aceleración} : fa(z) = 0 \tag{1.2.12}$$

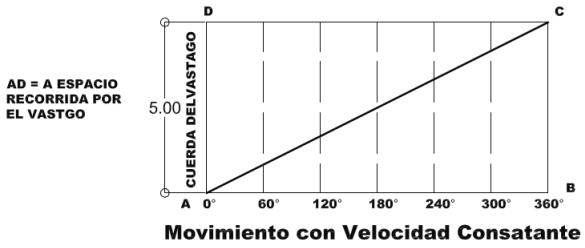


Figura 1.9. Diagrama de un movimiento lineal

1.2.4. Polinómico de 5^{to} grado, caso general

La ley de continuidad puede cumplirse estrictamente si se obliga a que el tramo que se pretende añadir conecte de forma precisa con sus adyacentes, es necesario asegurar la continuidad en las leyes de desplazamiento, velocidad y aceleración, lo que supone dos condiciones de contorno para cada una de ellas, una de las cuales se aplicará al comienzo del tramo y la otra al final; las condiciones serán las del tramo anterior y posterior respectivamente, de manera que se tienen seis condiciones de contorno que hay que satisfacer usando para ello un movimiento genérico versátil, para esto se emplean funciones polinomiales; en este

caso es suficiente un polinomio de 5^{to} grado genérico, pues deja seis incógnitas que resolver, y su cálculo se realiza para que la ecuación satisfaga las condiciones de contorno requeridas. Debido al carácter cíclico del movimiento, si el tramo polinómico es el primero que conforma el movimiento del seguidor, entonces el tramo anterior será el último, igualmente si el tramo polinómico es el último entonces el tramo posterior con el que debe enlazar será el primero(Serrano Muñoz, 2010). La ecuación de desplazamiento que crea un polinomio de 5^{to} grado viene dada por:

$$S = C_0 + C_1\left(\frac{\theta}{\beta}\right) + C_2\left(\frac{\theta}{\beta}\right)^2 + C_3\left(\frac{\theta}{\beta}\right)^3 + C_4\left(\frac{\theta}{\beta}\right)^4 + C_5\left(\frac{\theta}{\beta}\right)^5 \tag{1.2.13}$$

Los seis coeficientes son las incógnitas que deben determinarse imponiendo las condiciones de contorno al desplazamiento, la velocidad, y la aceleración al inicio y final del tramo polinómico; en este caso hay dos posibilidades, se pueden aplicar las condiciones de contorno tal y como se ha dicho u omitir el uso de la velocidad angular hasta el final y trabajar directamente con las derivadas de la función desplazamiento:

1.3. Ángulo de presión

El ángulo de presión determina la dirección de aplicación de la fuerza entre la leva y el seguidor, es decir, la normal común a ambas superficies con la dirección del movimiento de éste; de la descripción se deduce que para un seguidor de cara plana traslacional u oscilante, si la dirección de aplicación de la fuerza es normal a la superficie de contacto, entonces es perpendicular a su cara y coincide con la dirección de su movimiento, por lo que el ángulo de presión para este tipo de seguidores es nulo, es decir, la dirección de aplicación de la fuerza y la dirección de su movimiento son iguales; por eso no se considera en ellos el ángulo de presión como un parámetro cálculo a tener en cuenta (Norton, 2009). En los seguidores de rodillo, el punto que se toma para determinar la dirección del movimiento es su centro, el ángulo de presión ofrece una idea de la facilidad con la que la leva transmite el movimiento al seguidor: si es muy elevado, el seguidor puede atascarse o moverse con dificultad, mientras que en los oscilantes aumenta el rozamiento en el pivote sobre el que gira el brazo, lo que obliga a tratar de minimizarlo cuando se calcula una leva con un seguidor de rodillo. Existen desarrollos que plantean analíticamente el cálculo del ángulo de presión en función de magnitudes geométricas y de las curvas de desplazamiento y sus derivadas. No obstante, este mismo cálculo puede llevarse a cabo si se emplea el algoritmo planteado para generar el perfil,con lo que se obtendrá el ángulo de presión en función de los valores que va tomando el ángulo de rotación ϕ(ibíd.).

1.4. Métodos de obtención del perfil

Existen dos métodos de obtención del perfil de leva, el gráfico y el analítico. Muchos de las literaturas consultados hacen referencia a ambos métodos cuando explican el proceso de generación del perfil, dentro de ellos podemos citar a (Rohtbart,1956) otros autores exponen los métodos analíticos para determinar

gráficamente un perfil de leva, se realiza una inversión cinemática del mecanismo leva-seguidor, en la cual la leva se considera el elemento fijo y la guía o articulación del seguidor móvil. La inversión cinemática no afecta el movimiento relativo entre la leva y el seguidor. El perfil de la leva es la envolvente de las curvas correspondientes a las distintas posiciones(Shigley y Uicker, 1988).

1.5. Excentricidad

Excentricidad, este es un parámetro que seleccionado adecuadamente permite reducir las dimensiones de la leva sin afectar la eficiencia del mecanismo, manteniendo el ángulo de presión dentro de valores aceptables. Por definición, la excentricidad es la distancia mínima del centro del seguidor al diámetro vertical del círculo base; el valor de la excentricidad no puede superar ni ser igual que el radio primario (Radio base+Radio del seguidor) que puede ser positivo, negativo o cero: en el primer caso el centro del seguidor queda a la derecha (mirando hacia la pantalla) del eje vertical, si fuese negativo estaría situado a la izquierda; un valor positivo de excentricidad hará disminuir el ángulo de presión en el tramo de subida, pero lo que aumentará en el tramo de bajada. La excentricidad negativa hace lo contrario(Matthew y Delbert, 1976).

1.5.1. Aplicaciones de los mecanismos de levas

Entre las aplicaciones de los mecanismos de levas se pueden señalar los árboles de levas empleados para automatizar el cierre y apertura de las válvulas de admisión y escape en motores de combustión interna,otros casos son :

- Conmutadores.
- Cierre de puertas.
- Frenos de discos.
- Máquinas herramientas.

1.6. Módulos, componentes para el modelado y cálculo de los mecanismos de levas

Los mecanismos de leva son dispositivos mecánicos, por lo que los sistemas informáticos desarrollados para automatizar su diseño pueden considerarse de uso específico para la mecánica; existen diferentes soluciones, como hojas de cálculo y módulos incluidos en sistemas de diseño como Solid Edge y Autodesk Inventor; en las líneas que siguen se exponen algunas características de tales soluciones.

1.6.1. Autodesk Inventor

Es una herramienta de diseño asistido por computadora producida por la empresa Autodesk y salió al mercado por primera vez en 1999; es un modelador paramétrico de sólidos en 3D, que permite a los inge-

nieros y/o diseñadores crear prototipos digitales, gracias a un conjunto de funcionalidades para el diseño, la simulación, el análisis, la visualización, y generación de documentos; con esta herramienta se pueden integrar planos en 2D de AutoCAD crear representaciones digitales del producto final que les permite validar la forma, dimensiones, precisión del ensamble, así como el comportamiento del producto antes que sea construido(Inventor, 2017). Inventor utiliza formatos específicos para importar y exportar los diseños realizados como: archivo para las piezas (*.IPT*), ensamblajes (*.IAM*), vista de dibujo (*.IDW y .DWG*) y presentaciones (*IPN*); en las últimas versiones de Inventor se utiliza un gestor de formas (*Shape Manager*) como kernel de modelado geométrico el cual pertenece a Autodesk. En la versión profesional incluye herramientas para crear piezas de plástico y sus respectivos moldes de inyección, cuenta con análisis de tensión por elementos finitos y análisis dinámico, además de utilizar tecnologías como *iPart*, *iAssembly*, *iMates*, *iCopy*, *iLogic* hacen que el diseño sea más rápido y eficiente y su combinación con Autodesk Vault y Autodesk 360 la hacen líder en el mercado del diseño mecánico (ibíd.).

Solid Edge

Fue lanzado al mercado por primera vez en 1996 inicialmente desarrollado por *Intergraph*, actualmente pertenece y es desarrollado por *Siemens AG*; es un sistema paramétrico para el diseño de piezas y sistemas en tres dimensiones; permite el modelado de piezas de distintos materiales, doblado de chapas, ensamblaje de conjuntos, soldadura, funciones de dibujo en plano para ingenieros(Siemens PLM, 2018).

Solid Edge es una herramienta de *software* fácil de usar para el proceso de desarrollo de productos, simulación, fabricación y gestión del diseño, ya que combina la velocidad y simplicidad del modelado directo con la flexibilidad y el control del diseño paramétrico hecho posible con *synchronous technology*(ibíd.).

1.6.2. Aplicación interactiva tridimensional asistida por computadora

CATIA fue creado por la compañía *Dassault Systèmes*. El programa está desarrollado para proporcionar apoyo desde la concepción del diseño hasta la producción y el análisis de productos, está disponible para múltiples plataformas, *Microsoft Windows*, Solaris, IRIX y HP-UX.

En sus inicios fue desarrollado para servir en la industria aeronáutica. En la actualidad es ampliamente utilizado en la industria del automóvil para el diseño y desarrollo de componentes de carrocería. Aplicación interactiva tridimensional asistida por computadora (CATIA, por sus siglas en inglés) ofrece la posibilidad, no solo de modelar cualquier producto, sino de hacerlo en el contexto de su comportamiento en la vida real. Al parecer, de acuerdo a la información obtenida de los materiales revisados, el sistema CATIA no incluye la filosofía de acelerar el diseño de piezas, aunque sí posee facilidades para la construcción paramétrica de piezas y mecanismos complejos. En este caso se hace un poco complejo el diseño de mecanismos de leva, pues hay que realizarlo paso a paso siguiendo alguna metodología (Rossetti y Bampi, 1999).

SolidWorks

La empresa *SolidWorks Corporation* fue fundada en 1993 por Jon Hirschtick con su sede en Concord, Massachusetts y lanzó su primer producto, SolidWorks 95, en 1995. En 1997 *Dassault Systèmes*, conocida por su *software* CATIA adquirió la compañía.(SolidWorks, 2017)

SolidWorks es un modelador paramétrico de sólidos que permite modelar piezas y mecanismos, extraer de ellos tanto planos técnicos como información necesaria para la producción de componentes; el programa posee un grupo de productos con distintas funcionalidades(ibíd.):

- **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(ibíd.).
- **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(ibíd.).
- **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(ibíd.).
- **Gestión de datos de productos:** permite tener control sobre los datos de diseño y mejorar considerablemente la manera en que sus equipos administran y colaboran en el desarrollo de productos.
- **Diseño eléctrico:** proporciona una serie de funcionalidades de diseño de sistemas eléctricos para satisfacer las necesidades de los profesionales.

MITCalc

Es una aplicación de escritorio con funcionalidades de cálculo y diseño de sistemas mecánicos, puede usar diferentes estándares de cálculo como ANSI/AGMA 6022-C93 (Revisión de AGMA 341.02), ANSI/AGMA 6034-B92 (Revisión de ANSI/AGMA 6034-A87), DIN 3996, DIN 3975-1, DIN 3975-2, entre otros(MITCalc, 2017).

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

Permite la salida directa a diferentes sistemas de diseños propietarios, como:

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

- TurboCAD
- BricsCAD

FreeCad

Es un modelador 3D distribuido bajo licencia LGPL[1] es libre para todos sus usuarios, para la asistencia en ingeniería mecánica y el diseño de elementos mecánicos aunque también se emplea en una amplia gama de usos como la arquitectura. 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, Coin3D, una implementación del estándar Open Inventor, Qt y Python. También es multiplataforma y se ejecuta en los sistemas Windows, Linux / Unix y Mac OSX, con el mismo aspecto y las funcionalidades exactas en todas las plataformas (B. Falck, D. Falck y Collette, 2012). Tampoco FreeCad implementa la filosofía de los aceleradores de diseño.

1.6.3. Herramientas para el cálculo de mecanismos de leva

Con el objetivo de agilizar el trabajo de diseño se han desarrollado distintas herramientas que automatizan el proceso de cálculo de los mecanismos de levas. Algunas de estas son Dinacam y CamDas.

DinaCam

Es una herramienta la cual, dados los tiempos de elevación, bajada, reposo y la función de desplazamiento, modela el perfil de una leva; su principal desventaja es que el gráfico de la ley de desplazamiento es una imagen y no admite modificación.

CamDas

Es una herramienta desarrollada en visual Basic genera un perfil de levas teniendo en cuenta los parámetros radio base, tipo de seguidor,radio de curvatura y ángulo de presión, su principal ventaja es que permite modificar los diagramas de desplazamientos generados antes de construir la leva.

1.7. Metodología 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.

[1]La Licencia Pública General Reducida (LGPL, por sus siglas en inglés garantiza la libertad de compartir y modificar el software cubierto por ella, asegurando que el software

"Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decidió hacer una variación de la metodología AUP (Proceso Unificado Ágil), de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI(Rodriguez Sanchez, 2015)."

- **Fases:** La Metodología AUP variante UCI va a contener tres fases las cuales son inicio, ejecución, cierre porque así se adapta mejor al ciclo de vida de los proyectos de la UCI; para una mayor comprensión se muestra la figura A.1 del apéndice.

- **Disciplinas:** La AUP variante UCI propone siete disciplinas las cuales son modelado de negocio, requisitos, análisis y diseño, implementación, pruebas internas, pruebas de liberación y pruebas de aceptación; para la parte de gestión en la variante UCI se cubren con las áreas de procesos que define CMMI- DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto). Para una mayor comprensión se muestra la figura A.2 del apéndice.

- **Roles:** AUP propone 9 roles (Administrador de proyecto, Ingeniero de procesos, Desarrollador, Administrador de BD, Modelador ágil, Administrador de la configuración, *Stakeholder*, Administrador de pruebas, Probador), se decide para el ciclo de vida de los proyectos de la UCI tener 11 roles, manteniendo algunos de los propuestos por AUP y unificando o agregando otros. Para una mayor comprensión se muestra la figura A.2 del apéndice.

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:

- **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.

Se decide emplear la metodología AUP-UCI debido a que articula con las características del proyecto, además brinda bondades específicamente en la utilización del escenario cuatro; entre los elementos que

permiten tomar una decisión de aplicar el escenario cuatro, podemos mencionar que el negocio se encuentra bien definido, de igual forma el cliente estará acompañando a los miembros del equipo de desarrollo y finalmente la metodología permite que los requisitos funcionales puedan cambiar constantemente.

1.8. Herramientas y lenguajes para el desarrollo

En la actualidad debido a la evolución de las tecnologías informáticas y lenguajes de desarrollo, ha sido posible acelerar la realización de nuevas técnicas computacionales, que satisfagan las necesidades del hombre moderno, provocando que cada vez sea necesario el estudio de estas para la realización de un sistema determinado.

A continuación, se presenta una descripción de las herramientas y lenguajes empleados en la realización de este módulo.

1.8.1. Open Cascade Technology

La Tecnología Open Cascade (OCCT, por sus siglas en inglés), es un núcleo gráfico que proporciona funcionalidades y servicios para el modelado de superficies y modelado de sólidos en tres dimensiones, el intercambio de datos CAD y la visualización. La mayor parte de las funcionalidades de OCCT se encuentra disponible en forma de bibliotecas escritas C++. La biblioteca puede ser utilizada en el desarrollo de *software* de diseño y modelado 3D (CAD), fabricación (Fabricación Asistida por Computadora (CAM, por sus siglas en inglés)) o simulación numérica (Ingeniería Asistida por Computadora (CAE, por sus siglas en inglés)) . Es una tecnología de *software* libre que se puede distribuir y modificar bajo los términos de la Licencia Pública General de GNU (LGPL) versión 2.1, con excepción adicional(Chesbrough, 2006) .

Alternativamente, *Open CASCADE* puede ser utilizada bajo los términos de licencia comercial o acuerdo contractual. Está diseñada para ser altamente portátil y funciona en varias 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) .

Edición Comunitaria de Open Cascade (OCE, por sus siglas en inglés) 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/) .

El empleo en la presente investigación de *Open Cascade Community Edition* está determinado por que es el único núcleo gráfico de código abierto disponible.

1.8.2. Lenguaje C++

Es un lenguaje de programación diseñado a mediados de los años 80 por Bjarne Stroustrup. Su creación fue con el objetivo de extender al lenguaje de programación C con mecanismos que permitieran la manipulación de objetos. El nombre de C++ fue propuesto por Rick Mascitti en el año 1983 ya que hasta el momento se le conocía como C con clases.

"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. La librerías estándar de C++ puede ser implementada en el propio C++. Esto implica que C++ es suficientemente expresivo y eficiente para los sistemas de mayor demanda de tareas de programación. C++ es usado por millones de programadores en cada dominio de aplicación. Billones de líneas de C++ están actualmente desarrolladas. Muchos sistemas operativos poseen partes esenciales escritas en C++, como Windows, iOS, Linux, entre otros. Este lenguaje no fue diseñado con la computación numérica en mente, sin embargo, muchos programas de ingeniería, numéricos y científicos son realizados en C++. Este puede coexistir con código escrito en otro lenguaje. C++ es soportado por una variedad de librerías y conjuntos de herramientas, tales como Boost[2], QT, OpenCV[3], entre otras" (Stroustrup, 2013) . 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.

1.8.3. Marco de trabajo QT

Fue creado por *Trolltech* para el desarrollo de aplicaciones con las bibliotecas de QT las cuales son ampliamente usadas para desarrollar aplicaciones con interfaces gráficas de usuarios(Qt, 2015).

"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(ibíd.)." Qt está disponible bajo varias licencias: licencia comercial y para software libre con varias versiones de la GPL[4] y la LGPL. 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[5], tecnologías web, sensores, protocolos de comunicación (Bluetooth, puertos serie), XML[6] y procesamiento JSON[7], impresión, la generación de PDF, y mucho más"(ibíd.).

1.8.4. Visual Paradigm

Visual Paradigm es una herramienta Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés) 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 y la documentación

[2]Líbrerías portables.

[3]Procesamiento de imágenes en tiempo real

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

[5]Es un entorno de desarrollo para aplicaciones gráficas en 2D y 3D

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

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

de los programas. Esta herramienta emplea el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) (VisualParadim, 2017)..

Se caracteriza por:

- Disponibilidad en múltiples plataformas (Windows, Linux,MacOS).
- 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 decide su empleo para la confección de la propuesta de solución debido a que esta herramienta genera códigos en varios lenguajes, posee una licencia gratuita y emplea el lenguaje de modelado visual UML que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*.

1.8.5. Sistema de control de versiones

Git es un sistema de control de versiones distribuido, libre y de código abierto, diseñado para manejar proyectos pequeños o muy grandes con rapidez y eficiencia(GitLab, 2018). La característica Git que realmente hace que sea parte de casi todas Administración de Código Fuente (SCM, por sus siglas en inglés) es su modelo de ramificación que permite tener múltiples ramas locales que pueden ser totalmente independientes entre sí. La creación, la fusión y la supresión de esas líneas de desarrollo toma segundos. Esto significa que se pueden ejecutar acciones como las indicadas en:

- **Conmutación de contexto sin fricción:** Crear una rama para probar una idea, comience varias veces, cambie de nuevo a la rama de donde se ramificó, aplique un parche, cambie de nuevo a la rama donde está experimentando y fijarlo .
- **Reglas Basadas en el Rol:** Tener una rama que siempre contiene solo lo que va a la producción, otra que se fusiona en el trabajo para la prueba, y varias más pequeñas para el trabajo diario.
- **Flujo de trabajo basado en funciones:** Crear nuevas ramas para cada nueva función en la que esté trabajando, de modo que pueda cambiar sin problemas entre ellas y, a continuación, suprimir cada una de las ramas cuando se fusione en su línea principal.
- **Experimentación desechable:** Crear una rama para experimentar, darse cuenta de que no va a funcionar, y solo eliminarlo, abandonar el trabajo.

Se elige Git-Lab como sistema de control de versiones durante el desarrollo de la propuesta de solución, debido a las características antes mencionadas y por indicaciones del proyecto al que está integrado la pre-

sente investigación.

1.9. Conclusiones del capítulo

Entre las conclusiones parciales a que hemos arribado en el presente capítulo tenemos:

1. El estudio de las herramientas homólogas permitió comprender mejor el proceso de modelado de los mecanismos de leva.
2. El estudio de la secuencia de diseño de los mecanismos de leva permitió obtener una forma viable y efectiva para acelerar el proceso de diseño de mecanismos de leva.

CAPÍTULO 2

Propuesta de Solución

En el presente capítulo se aborda las cuestiones referentes a la propuesta de solución, la cual incluye un mapa conceptual del componente, la descripción del componente, los requisitos funcionales y no funcionales las historias de usuario y los aspectos particulares del diseño, como: el estilo y patrón arquitectónico, los patrones de diseños, el modelo de clases.

2.1. Descripción del componente de la solución propuesta

El módulo tiene un diseño en el que están consideradas las funcionalidades necesarias para modelar mecanismos de leva de diferentes tipos (lineal, cilíndrica y disco) y puede realizar los cálculos asociados a su diseño en correspondencia con las normas establecidas y otras fuentes literarias. La ventana de diseño posee un grupo de elementos que determinan el modelado de los mecanismos, entre los que se puede resaltar la función de desplazamiento por tramos, de igual forma se establecen las dimensiones como ancho, radio de la leva y del seguidor,propiedades como excentricidad y ángulo de presión entre otros; también posee un botón "*Cancel*" para cerrar la ventana. La ventana tiene un botón "*Calculate*" que se encarga de realizar los procedimientos matemáticos del componente desde el punto de vista dinámico; hay una sección en la que se muestran mensajes de errores de diseño; el botón "*aceptar*" al activarse cierra la ventana y se realiza el modelado que tiene su representación en el visor. La función de desplazamiento determina el perfil de leva las mismas están definidas en apartado Funciones matemáticas que definen la ley de desplazamiento, a partir de la cantidad de tramos seleccionados los cuales pueden tener un mínimo de tres y un máximo de doce tramos se pueden obtener diferentes configuraciones del perfil deseado pudiendo obtener hasta treinta y seis combinaciones básicas de levas .

2.2. Mapa Conceptual

El mapa conceptual no se está utilizando desde el punto de vista ingenieril, sino para apoyar la descripción de la solución, por lo tanto, el mismo puede ser entendido por cualquier persona y en él se van a

involucrar conceptos ya sea del sistema o del dominio 2.1 (Ballester Vallori, 2005).

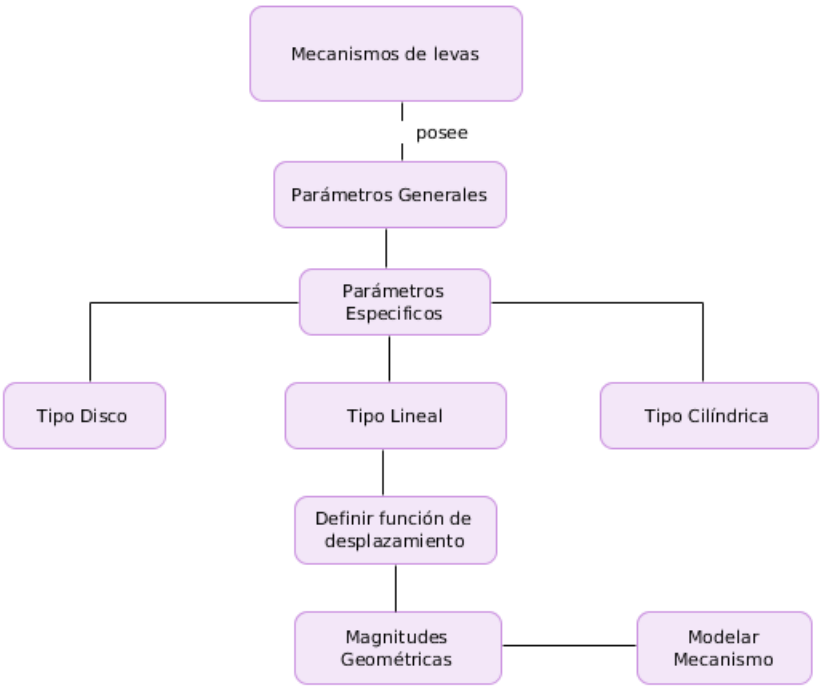


Figura 2.1. Mapa Conceptual

2.3. Requisitos

Los requisitos de un sistema son la descripción de los servicios proporcionados por el mismo y sus restricciones operativas, estos 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 (Sommerville, 2006). 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.

2.3.1. 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 (ibíd.). A continuación, se enumeran los requisitos funcionales definidos para el módulo:

RF 1. Seleccionar mecanismo de leva según su tipo.

RF 2. Insertar los parámetros generales según el tipo de mecanismos.

RF 3. Insertar los parámetros específicos del mecanismo de leva de tipo disco.

RF 4. Insertar los parámetros específicos del mecanismo de leva de tipo lineal.

RF 5. Insertar los parámetros específicos del mecanismo de leva de tipo cilíndrico.

RF 6. Mostrar el gráfico de desplazamiento del resultado de los cálculos de los mecanismos de leva.

RF 7. Mostrar los errores al diseñar el componente.

RF 8. Modelar el mecanismo diseñado.

2.3.2. Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema, incluyen restricciones de tiempo, sobre el proceso de desarrollo y normas o estándares, además, no se aplican regularmente a rasgos o servicios individuales del sistema(Sommerville, 2006).

Funcionalidad:

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

Portabilidad:

Poder ser ejecutado en diferentes sistemas operativos basados en GNU Linux de 64-bits

2.4. 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(Jeffries, Anderson y Hendrickson, 2001). A continuación se muestra la Historia de Usuario “Seleccionar mecanismos de leva según tipo”. Para consultar las restantes Historias de Usuario remitirse al Anexo A.2.

Tabla 2.1. Historia de usuario #1

Historia de usuario

Continúa en la próxima página

Tabla 2.1. Continuación de la página anterior

Número: 1	**Nombre:**Seleccionar mecanismo de leva según su tipo
Programador: Reydel Baños Acosta	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 hora
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 0.5 horas

Descripción:

1- Objetivo: Permitir la selección del mecanismo de leva según su tipo.

2- Acciones para lograr el objetivo (precondiciones y datos): Para la selección del mecanismo de leva según su tipo es necesario:

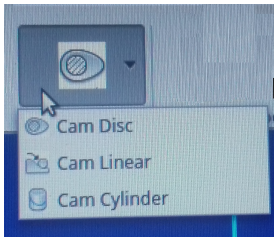
- Seleccionar el "*ComboBox*" correspondiente al mecanismo de leva según su tipo que se desee visualizar.

3- Flujo de la acción a realizar: 3- Flujo de la acción a realizar:

- El usuario debe seleccionar el "*ComboBox* DisCam.
- Si se selecciona el botón "*Cancel*" se cierra la ventana.

Observaciones: Permitir la selección del mecanismo de leva .

Prototipo elemental de interfaz gráfica de usuario



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 (Sommerville, 2006). El proceso de diseño tiene asociada la decisión del tipo de 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.

2.6. 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 exis-

tente se vaya a someter a reingeniería, la imposición de un estilo arquitectónico desembocará en cambios fundamentales en la estructura del software, incluida una resignación de la funcionalidad de los componentes (Pressman, 2005). Se escoge como estilo arquitectónico el de Llamada y Retorno, pues permite que un diseñador de software obtenga una estructura de programa que resulta fácil de modificar y cambiar de tamaño (ibíd.). 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.

2.6.1. Arquitectura en Dos Capas

La arquitectura se encuentra organizada mediante un sistema de dos capas,la de presentación y lógica del negocio, las cuales proporcionan un conjunto de servicios. Este modelo soporta el desarrollo incremental del sistema, los cambios y la portabilidad. Además, cuando las interfaces de las capas cambian o se añaden nuevas funcionalidades a una capa, solamente se ven afectadas las adyacentes; entre las desventajas de este modelo se encuentra 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. Se decide el empleo de esta arquitectura en la confección del módulo porque soporta bien los cambios y el desarrollo incremental.

El diseño arquitectónico es el más apropiado para el desarrollo de las funcionalidades para el modelado de mecanismo de levas, debe basarse en lo fundamental, en una Arquitectura en capas que permita la comunicación con el núcleo de la aplicación y con otros módulos de la misma.

2.6.2. 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.

Los patrones Patrones Generales de Software para Asignación de Responsabilidades (Grasp, por sus siglas en inglés) definidos en las funcionalidades son:

- **Experto**: mediante su uso, se asignan responsabilidades a la clase que cuenta con la información necesaria,(Larman, 2003) este patrón se evidencia en las clases que se encargan de la creación de los mecanismos.
- **Creador**: permite crear objetos de una clase determinada,(ibíd.) este patrón se evidencia en las clases de interfaz DlgCamshaft ya que tiene la información necesaria para la creación de objetos tipo CamShaft.
- **Polimorfismo**: se emplea para asignar comportamientos distintos según el tipo de clase, aquí se evidencia en las clases, que heredarán de la clase Camshaft y reimplementan los métodos execute, onChanged, mustExecute.

- **Alta cohesión**: este patrón caracteriza a las clases que posean responsabilidades estrechamente relacionadas, es decir, que no realicen un trabajo enorme (Larman, 2003).
- **Bajo acoplamiento**: posibilita que una clase no dependa mucho de otras clases (ibíd.).

Los patrones Gang of Four (GOF, por sus siglas en inglés) definido en las funcionalidades son:

- **Observador:**Construye una dependencia entre un sujeto y sus observadores de modo que cada modificación del sujeto sea notificada a los observadores para que puedan actualizar su estado (ibíd.). Este patrón se evidencia con la función "*mustExecute*" la que permite que las propiedades de las piezas modeladas, puedan ser editadas.

2.6.3. 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, métodos, atributos, y las relaciones entre los objetos (herencia, agregación, asociación, entre otras) ver en la figura 2.2

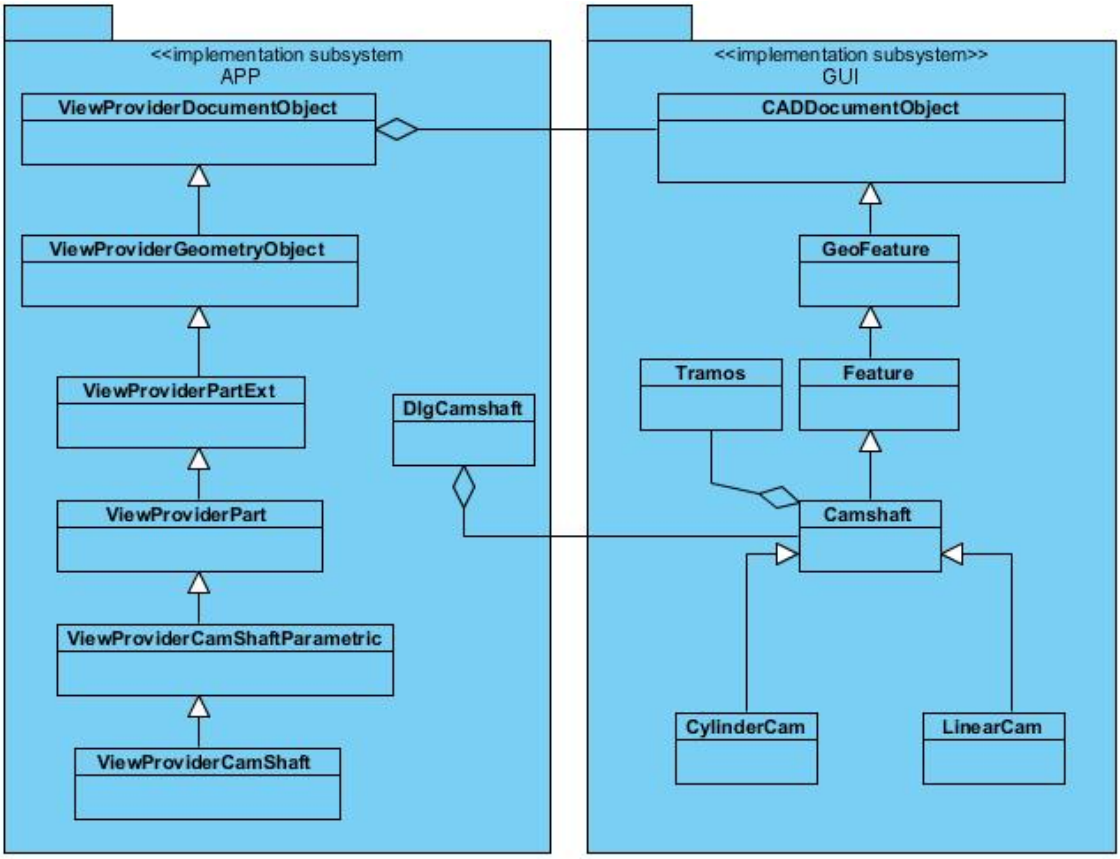


Figura 2.2. Diagrama de Clases

2.7. Conclusiones del capítulo

- A partir del estudio del marco teórico se definió una propuesta de solución viable.
- El diseño de los artefactos obtenidos en este capítulo sientan las bases para dar inicio al próximo capítulo, el cual corresponde a la implementación y prueba de la solución propuesta.

CAPÍTULO 3

Implementación y Prueba

En el presente capítulo se abordan cada uno los componentes que integran la etapa de implementación y prueba del módulo 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 implementación es la elaboración de una aplicación o ejecución de un plan, idea o algoritmo en donde el usuario se involucra en su desarrollo. La fase de implementación del software posee como entradas los artefactos de 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 en el código 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 establecido en el grupo de investigación al que se encuentra insertada la propuesta de solución.

Descripción	Ejemplo
Definición de Objetos, Clases, funciones y atributos	

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

Siempre se declara para todas las clases implementadas su respectivo destructor de clase.	virtual ~Foo()
La declaración de funciones o métodos siempre comenzarán en letra inicial minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase.	<Tipo dato retorno> funcion() <Tipo dato retorno> funcionCompuesta() <Tipo dato retorno>funcionDobleCompuesta()
Los atributos siempre estarán escritos con letra minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase.	<Tipo dato> atributo; <Tipo dato> atributoNombreCompuesto;
Definición de parámetros dentro de las funciones y constructores de clases	
Los nombres de los identificadores de los parámetros en las funciones deben estar escritos con minúscula separados a un espacio cada uno y en caso de ser compuesto utilizar normativa CamelCaselowerCamelCase.	<Tipo dato retorno> funcion(<tipo><id1>, <tipo><id2>, <tipo><idN>)
Los identificadores de los parámetros dentro de los constructores de las clases deben estar escritos con minúscula separados a un espacio cada uno y en caso de ser compuesto utilizar normativa CamelCase-lowerCamelCase.	Clase(<tipo><id1>, <tipo><id2>, <tipo><idN>)
Definición de expresiones	

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

Definición de estructuras de control y bucles	

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

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

Figura 3.1. Estándares de codificación

3.1.2. Diagrama de Componente

Un diagrama de componentes representa como es dividido en componentes un sistema de *software*; 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.

Estos diagramas pueden contener paquetes que permiten organizar la construcción del sistema de información en subsistemas y que recogen aspectos prácticos relacionados con la secuencia de compilación entre componentes, la agrupación de elementos en librerías(Cillero, 2018).

En la Figura 3.2 se muestra el diagrama de componentes de la solución, este cuenta con los paquetes App, encargado de contener los componentes que integran las clases encargadas del manejo de los datos y la lógica, Gui, es el que abarca los componentes relacionados con las interfaces de usuario, y el paquete Qt que contiene todas las librerías que nos brinda el framework Qt.

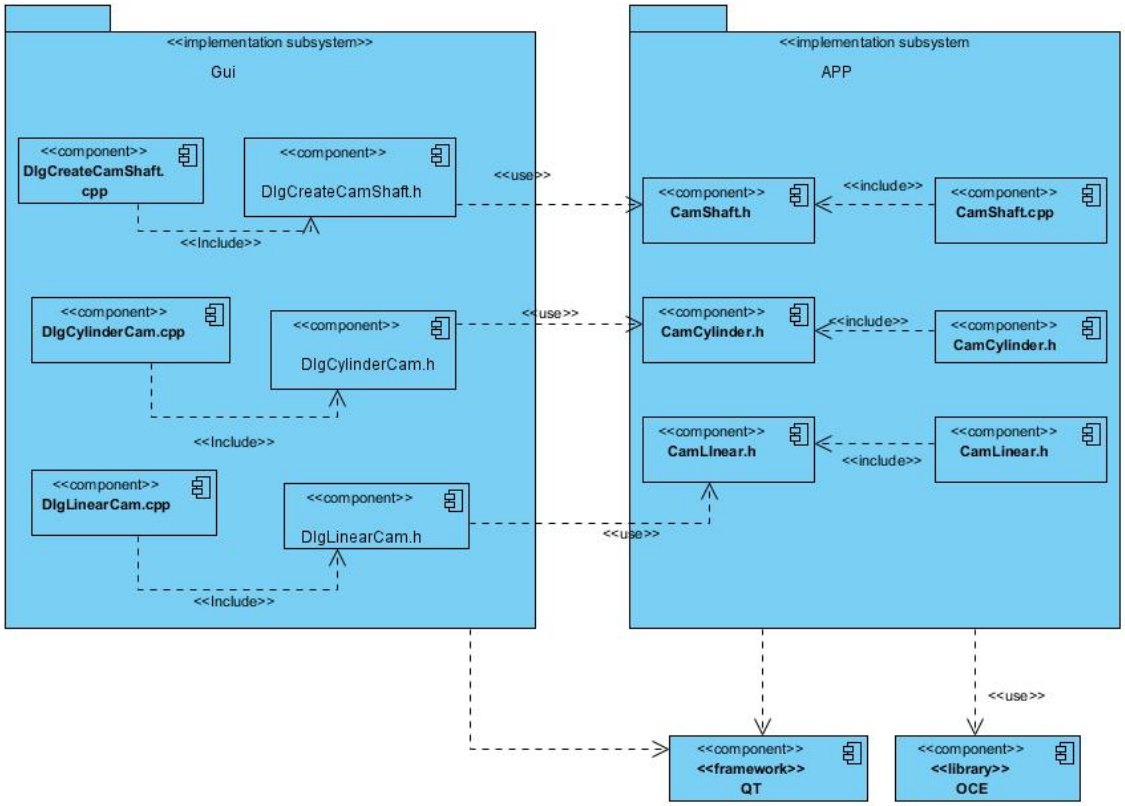


Figura 3.2. Diagrama de Componente

Construcción de los mecanismos de levas

Creando el mecanismo de leva de tipo disco

Para iniciar la construcción del mecanismo es necesario representar una circunferencia de construcción:

- **Circunferencia de referencia:** Es la curva de construcción fundamental; se crea posicionando un punto como centro, en este caso el punto (0,0,0).
- **Ley o función de desplazamiento :** Determina el perfil de la leva la misma se ve determinado a partir de las funciones explicadas en el apartado Funciones matemáticas que definen la ley de desplazamiento
- **Parámetros geométricos:** Definen la configuración geométrica de la leva, los más importantes son el ángulo de presión, la excentricidad, así como los puntos inicial y final del movimiento.

El mecanismo se construye a partir de la circunferencia de referencia utilizando la topología wire que nos brinda Open CASCADE, luego en un segundo momento se convierte esta topología en Face utilizando la funcionalidad de Open CASCADE TopoDS_Face y por último paso se construye un sólido como se muestra en la figura 3.3:

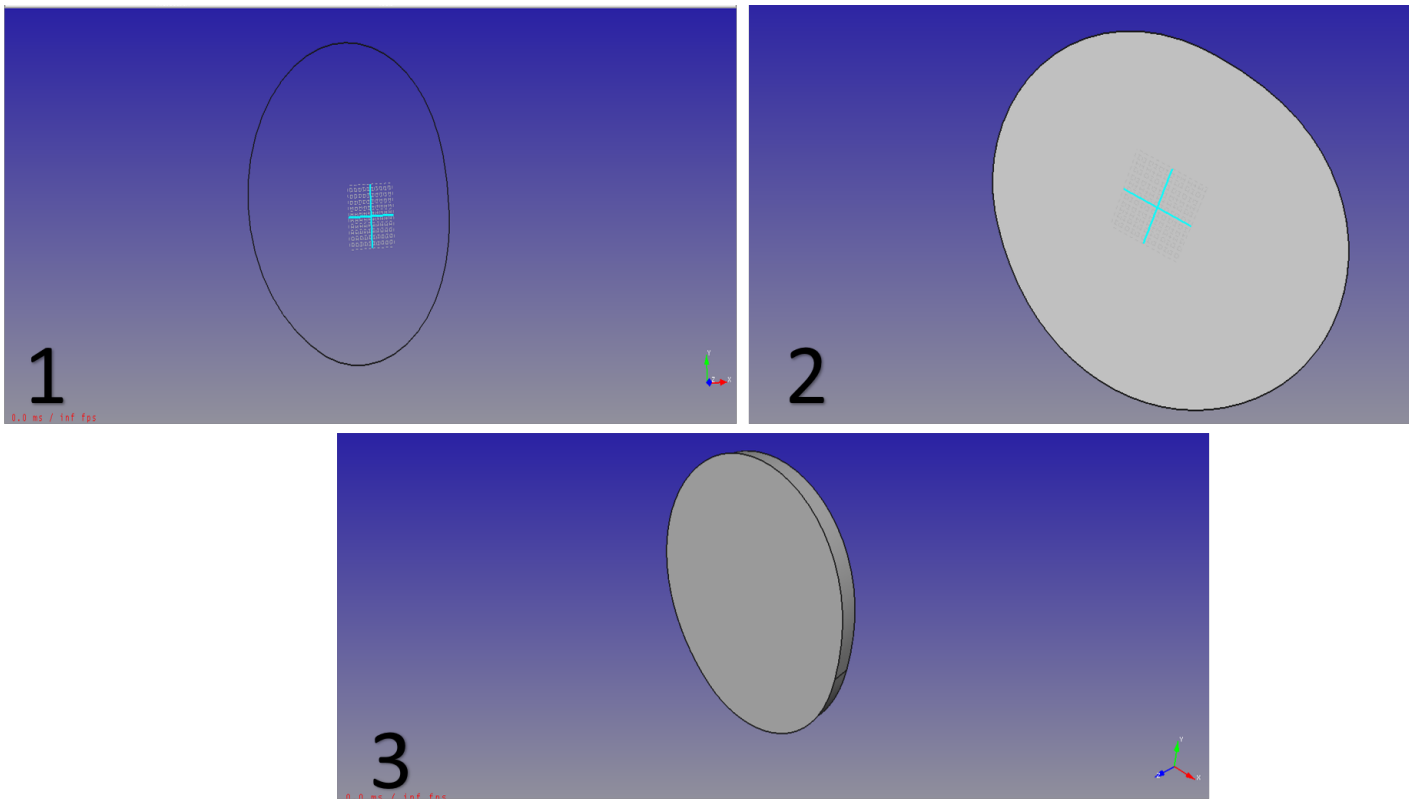


Figura 3.3. Secuencia de imágenes del perfil de un mecanismo de leva de tipo disco.
fuente:elaboración propia

Creando el mecanismo de leva de tipo lineal

- **Segmentos:** Se construye a partir de segmentos con la topología wire.
- **Ley o función de desplazamiento :** La ley o función de desplazamiento determina el perfil de la leva la misma se define a partir de las funciones explicadas en el apartado Funciones matemáticas que definen la ley de desplazamiento
- **Parámetros geométricos:** Para la construcción del mecanismo son necesarios también los parámetros geométricos que lo definen como ángulo de presión, excentricidad, final del movimiento entre otras.

El mecanismo se construye a partir del punto P(0,0,0) utilizando la topología wire que nos brinda Open CASCADE, luego en un segundo momento se convierte esta topología en Face utilizando las funcionalidad de Open CASCADE TopoDS_Face y por último paso se construye un sólido como se muestra en la figura 3.4:

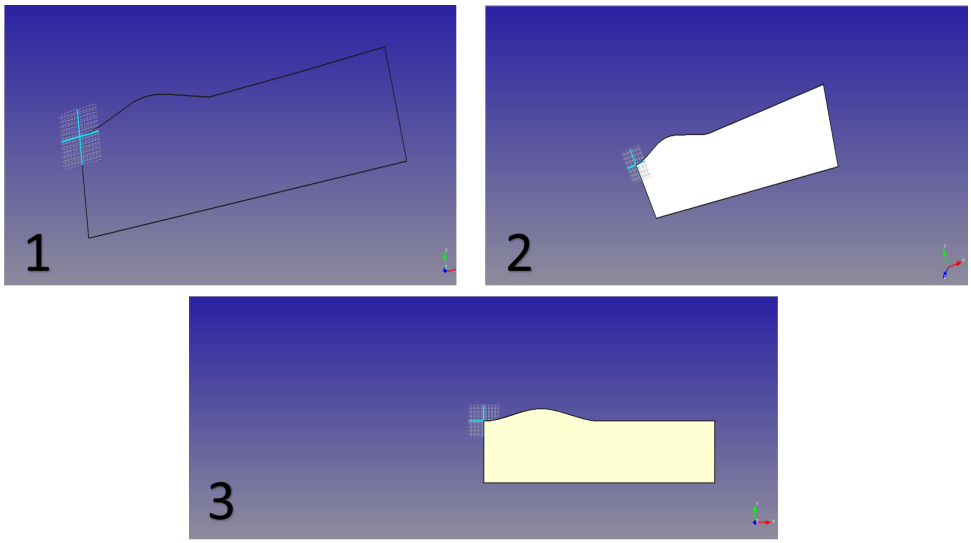


Figura 3.4. Secuencia de imágenes del perfil de un mecanismo de leva de tipo lineal.
fuente:elaboración propia

Creando el mecanismo de leva de tipo cilíndrico

- **Cilindro:** Se construye a partir de la primitiva cilindro que brinda Open CASCADE utilizando la ecuación del cilindro.
- **Segmentos:** Se construye el camino a partir de la funcionalidad de Open CASCADE TopoDS_wire.
- **Ley o función de desplazamiento :** La ley o función de desplazamiento determina el perfil de la leva la misma se define a partir de las funciones explicadas en el apartado Funciones matemáticas que definen la ley de desplazamiento.

El mecanismo se construye a partir de realizar un corte al cilindro con un camino descrito por la función de desplazamiento conociendo con anterioridad las ecuaciones parámetricas del cilindro como se puede observar en la figura 3.5:

$$x = r * cos(\Theta) \tag{3.1.1}$$

$$y = r * sin(\Theta) \tag{3.1.2}$$

Donde:

- x : Ordenada x del punto del cilindro.
- y : Ordenada y del punto del cilindro.
- r : Radio del cilindro.

El mecanismo se construye a partir de realizarle un camino al cilindro definido por la función de desplazamiento como se muestra en la figura:

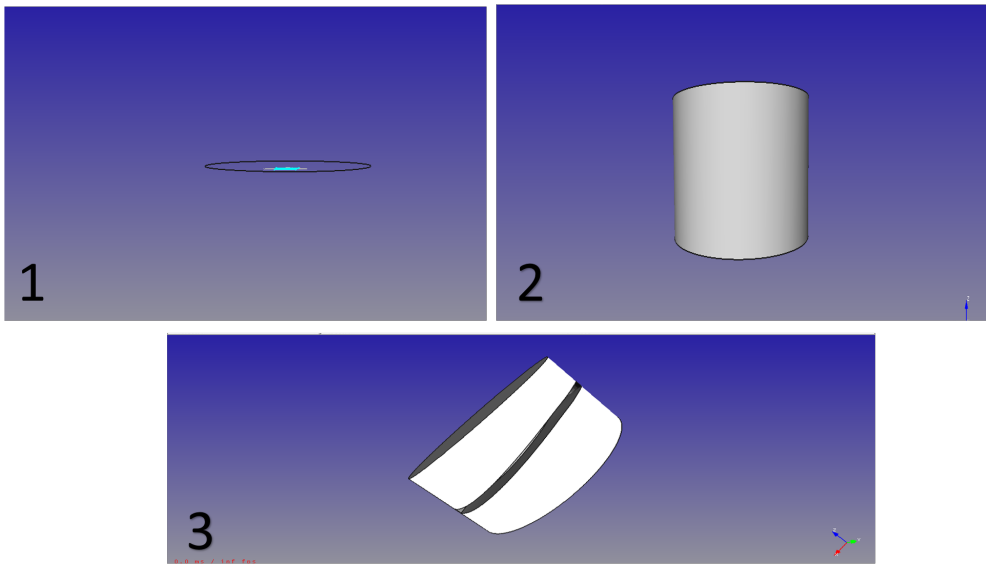


Figura 3.5. Secuencia de imágenes del perfil de un mecanismo de leva de tipo cilíndrico.
fuente:elaboración propia

3.2. Pruebas

El proceso de pruebas de software tiene los siguientes objetivos (Sommerville, 2006):

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

3.2.1. Niveles de pruebas

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

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

- Prueba de Integración: es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es tomar los módulos probados mediante la prueba unitaria y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.
- Pruebas de Sistema: está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas. Algunas de estas son: pruebas de recuperación, seguridad, resistencia, entre otras.
- Pruebas de Aceptación: es la realización de una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. Un plan de prueba traza la clase de pruebas que se han de llevar a cabo, y un procedimiento de prueba define los casos de prueba específicos en un intento por descubrir errores de acuerdo con los requisitos.

3.2.2. Método de prueba

El principal objetivo del diseño de casos de prueba es obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software. Para llevar a cabo este objetivo, se emplearán los dos métodos de prueba (Ruiz Tenorio, 2010).

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

3.2.3. Detalles técnicos de la implementación

Objetos y funciones empleadas de Open CASCADE:

- **gp_Ax2d:** Crea un eje representando en el eje X un objeto de referencia en el sistema de coordenadas.
- **gp_Circ2d:** Crea un círculo en dos dimensiones en el plano. Este circulo esta definido por un radio y una posición en el plano.
- **gp_Pnt:** Crea un punto en tres dimensiones.
- **gp_Pnt2d:** Crea un punto en dos dimensiones.
- **GCE2d_MakeSegment:** Crea un segmento entre dos puntos en dos dimensiones.
- **GCE2d_MakeCircle:** Crea un círculo en el espacio a partir de uno en el plano.

- **TopoDS_Wire:** Topología presente en la biblioteca Open CASCADE crea un segmento a partir de bordes.
- **TopoDS_Face:** Topología presente en la biblioteca de Open CASCADE crear un face de un *wire* cerrado.
- **TopoDS_Shape:** Topología presente en la biblioteca de Open CASCADE, crea un sólido.
- **TopoDS_Compound:** Función que permitir unir varios *shape* para poder trabajar con ellos.
- **BRepAlgoAPI-Cut:** Función que realiza una operación *boolean* para realizar un corte sobre dos sólidos.
- **BRepBuilderAPI_MakeWire:** Topologías de la biblioteca Open CASCADE crea un segmento.
- **BRepBuilderAPI_MakeEdge:** Topologías de la biblioteca Open CASCADE, crea un borde.
- **BRepFill_PipeShell:** Topologías de la biblioteca Open CASCADE,realiza el *Swep* lo que es equivalente a una obstrucción.

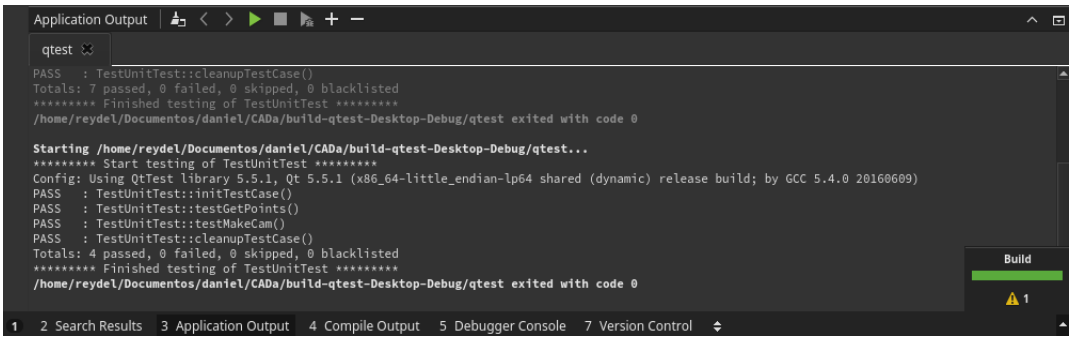
3.2.4. Diseño de caso de pruebas

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, fueron realizadas 3 iteraciones ver en los anexos Tablas de casos de pruebas .

3.2.5. Prueba unitarias

Para la realización de las pruebas unitarias se utiliza el"***Qt Test***", un *framework* para realizar pruebas unitarias a aplicaciones y bibliotecas basadas en Qt. El"***Qt Test***" proporciona todas las funcionalidades comúnmente encontradas en los *framework* de pruebas, así como extensiones para probar interfaces gráficas de usuario(QtTest, 2018). Los resultados de la ejecución de las pruebas se muestran en figura 3.6.

Figura 3.6. Iteración con ***Qt Test***



3.2.6. Resultados de las pruebas

La realización de pruebas funcionales permitió identificar la presencia de no conformidades, las cuales se listan en la tabla 3.1.

Tabla 3.1. No conformidades detectadas en las pruebas

No. NC	Requisito Funcional	Descripción	Complejidad	Estado
1	RF 1	No permite la selección del tipo de mecanismo correctamente para el modelado	Media	Resuelta
2	RF 4	Los "*radioButtom*" no permiten la selección	Alta	Resuelta
3	RF 6	El gráfico de la función de desplazamiento no se muestra.	Media	Resuelta
4	RF 5	No se actualizan lo valores en la leva de tipo cilíndrica al cambiar un valor.	Baja	Resuelta
5	RF 7	Si hay errores en algunos de los campos, si modifico los valores a otros correctos, siguen marcándose en color rojo.	Baja	Resuelta
6	RF 5	No se muestran las unidades de medidas.	Baja	Resuelta

Al concluir cada una de las iteraciones planificadas para el desarrollo de la propuesta de solución, fueron realizadas las pruebas pertinentes. En la figura 3.7 se muestra el resultado obtenido:

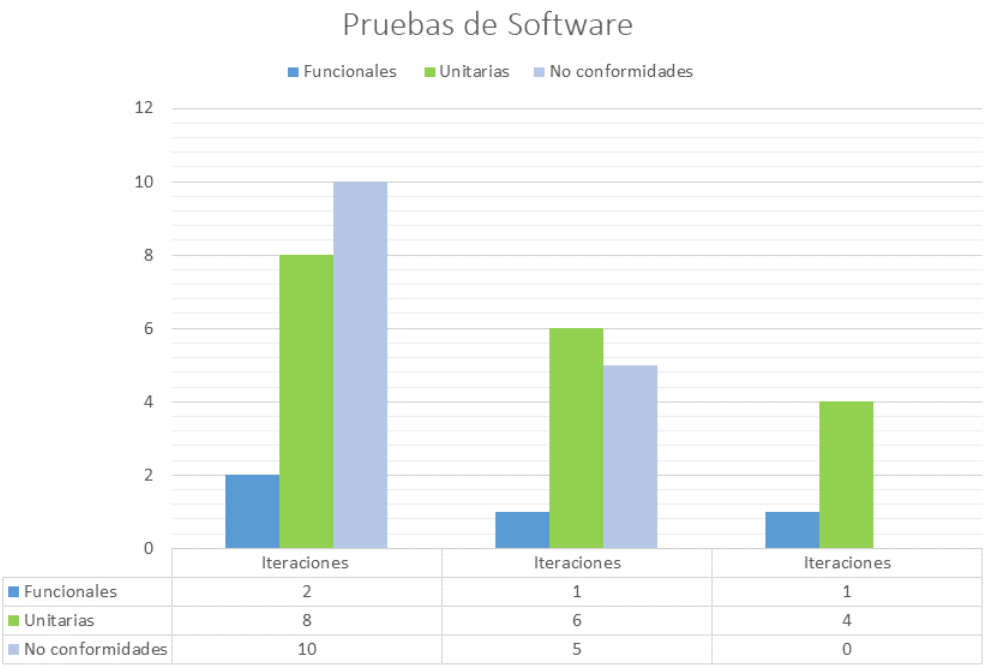


Figura 3.7. Resultados de las pruebas aplicadas al *software* en las distintas iteraciones

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

3.3. Conclusiones del capítulo

- La utilización de la técnica de caja negra, permitió identificar las no conformidades, las cuales fueron corregidas garantizando una mejor calidad de la solución.
- La realización del diagrama de componente, permitió dividir el componente en partes modulares, reemplazables y sustituible que encapsularan las implementaciones del software.

Conclusiones

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

1. Con el módulo desarrollado se acelera el proceso de diseño de tres tipos de levas (de disco, lineal y cilíndrica) reduciendo de esta manera el tiempo de modelado con respecto a los procedimientos tradicionales.
2. Las funcionalidades implementadas permiten modelar variantes de perfiles para tres tipos de levas, basados en ocho funciones típicas, garantizando una amplia gama de combinaciones para el diseño.
3. El resultado contribuye al fortalecimiento de la soberanía tecnológica en el área del diseño asistido por computadoras aplicado a la mecánica, pues está basado en sistemas de código abierto.

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:

- Agregar las funcionalidades requeridas para el modelado de árboles de levas a partir de los perfiles diseñados con el módulo.
- Incluir los cálculos de resistencia del material de acuerdo a diversas normas.
- Desarrollar un sistema de base de datos para la aplicación portadora del módulo que garantice la selección de diferentes materiales para la leva y el seguidor, así como cargar perfiles de leva normalizados

Glosario

LGPL (Es un tipo de licencia que garantiza la libertad de compartir y modificar el software cubierto por ella, asegurando que el software es libre para todos sus usuarios.). 19, 21, 22

Acrónimos

CAD Diseño Asistido por Computadora. 3, 18, 21
CAE Ingeniería Asistida por Computadora. 21
CAM Fabricación Asistida por Computadora. 21
CASE Ingeniería de Software Asistida por Computadora. 22
CATIA Aplicación interactiva tridimensional asistida por computadora. 17, 18

GOF Gang of Four. 30
GPL Licencia Pública General. 22
Grasp Patrones Generales de Software para Asignación de Responsabilidades. 29
GUI Interfaces Gráficas de Usuario. 22

JSON Notación de Objetos de JavaScript. 22

OCCT La Tecnología Open Cascade. 21
OCE Edición Comunitaria de Open Cascade. 21

SCM Administración de Código Fuente. 23

UML Lenguaje Unificado de Modelado. 23

XML Lenguaje Extensible de Enmarcado. 22

Referencias bibliográficas

[1] Araworks. *Ara Works. Precios para compra*. 2018. URL: http://araworks.es/precios-productos-solidworks/ (vid. pág. 2).

[2] Asidek. *AutoCad 2019*. 2018. URL: https://www.asidek.es/producto/autocad-2019/ (vid. pág. 2).

[3] Asidek. *Autodesk Inventor 2017*. 2018. URL: https://www.asidek.es/producto/autodesk-inventor-20 (vid. pág. 2).

[4] ASIXMEC. «Aplicación Cubana para el diseño asistido por computadora». En: (). URL: http://www.3dcadportal.com/asixmec-aplicacion-cubana-para-el-diseno-e-ingenieria-asistida-por-computadora.html (vid. pág. 1).

[5] Antoni Ballester Vallori. «El aprendizaje significativo en la práctica». En: *V Congreso Internacional Virtual de Educación*. 2005 (vid. pág. 26).

[6] Salvador Cardona y Daniel Clos Acosta. *Teoría de Máquinas (Edición en Castellano)*. 2001 (vid. págs. 8, 11).

[7] Henry William Chesbrough. *Open innovation: The new imperative for creating and profiting from technology*. Harvard Business Press, 2006 (vid. pág. 21).

[8] Manuel Cillero. *Diagrama de componentes*. 2018. URL: https://manuel.cillero.es/doc/metrica-%203/tecnicas/diagrama-de-componentes (vid. pág. 34).

[9] Victor M Franco Correia et al., «Modelling and design of adaptive composite structures». En: *Computer Methods in Applied Mechanics and Engineering* 185.2-4 (2000), págs. 325-346 (vid. pág. 9).

[10] Brad Falck, Daniel Falck y Brad Collette. *Freecad [How-To]*. Packt Publishing Ltd, 2012 (vid. pág. 19).

[11] Juan Manuel Marín García. *Apuntes de diseño de máquinas*. Editorial Club Universitario, 2008 (vid. pág. 11).

[12] GitLab. *GItLab*. 2018. URL: https://about.gitlab.com/ (vid. pág. 23).

[13] Edward J Haug. *Computer aided kinematics and dynamics of mechanical systems*. Vol. 1. Allyn y Bacon Boston, 1989 (vid. pág. 12).

[14] Autodesk Inventor. *Autodesk Inventor Profesional*. Mayo de 2017. URL: http://www.imaginit.com/software/autodesk-products/inventor (vid. pág. 17).

[15] Ron Jeffries, Ann Anderson y Chet Hendrickson. *Extreme Programming Installed*. Ed. por Addison-Wesley Professional. 2001 (vid. pág. 27).

[16] Craig Larman. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado*. 2da edición. Pearson Educación, 2003 (vid. págs. 29, 30).

[17] Gary K.and Matthew y Tesar Delbert. *The Dynamic Synthesis, Analysis, and Design of Modeled Cam Systems*. 1976 (vid. pág. 16).

[18] MITCalc. *MITCalc - Mechanical, Industrial and Technical Calculations*. 2017. URL: http://www.mitcalc.com/ (vid. pág. 18).

[19] José Luis Montes de Oca Montano. «La migración hacia software libre en Cuba: complejo conjunto de factores sociales y tecnológicos en el camino de la soberanía nacional». En: *Revista Universidad y Sociedad* 7.3 (2015), págs. 119-125 (vid. pág. 1).

[20] Robert L. Norton. *Diseño de Maquinaria*. 4ta edición. McGraw-Hill, 2009 (vid. págs. 1, 15).

[21] Siemens PLM. *Siemens. Ingenuity for live*. 2018. URL: https://www.plm.automation.siemens.com/store/en-us/solid-edge/ (vid. pág. 17).

[22] Simiens PLM. *Solid Edge*. 2018. URL: https://www.plm.automation.siemens.com/store/en-us/solid-edge/ (vid. pág. 2).

[23] Roger S. Pressman. *Ingeniería del software. Un enfoque práctico*. 6ta Edición. McGraw-Hill, Nueva York, 2005 (vid. pág. 29).

[24] About Qt. *About Qt - Qt Wiki*. 2015. URL: https://wiki.qt.io/About_Qt (vid. pág. 22).

[25] QtTest. *QtTest*. 2018. URL: http://doc.qt.io/qt-5/qtest-overview.html (vid. pág. 40).

[26] Tamara Rodriguez Sanchez. «Metodología de desarrollo para la actividad productiva de la UCI». Tesis de mtría. 2015 (vid. pág. 20).

[27] Rosaldo José Fernandes Rossetti y Sergio Bampi. «A software environment to integrate urban traffic simulation tasks». En: *Journal of Geographic Information and Decision Analysis* 3.1 (1999), págs. 56-63 (vid. pág. 17).

[28] Harold A. Rothbart. *Cam Design Handbook*. Ed. por Harold A. Rothbart. McGraw-Hill, 2004 (vid. págs. 1, 5, 11, 13, 14).

[29] Roberto Ruiz Tenorio. *Las Pruebas de Software y su Importancia en las Organizaciones*. 2010 (vid. págs. 38, 39).

[30] Alejandro Marco Serrano Muñoz. «Diseño, análisis y simulación de levas planas : aplicación informática de CAM-DAS». Tesis de mtría. Universidad Carlos III de Madrid, 2010 (vid. págs. 6, 12, 15).

[31] Joseph Shigley y John Uicker. *Teoría de Máquinas y Mecanismos*. McGraw-Hill, 1988 (vid. pág. 16).

[32] SolidWorks. *SolidWorks_Corporation*. 2017. URL: `http://www.solidworks.com.mx` (vid. pág. 18).

[33] Ian Sommerville. *Ingeniería de Software*. 8va Edición. 2006 (vid. págs. 26-28, 38).

[34] Bjarne Stroustrup. *The C++ Programming Lenguage*. 4th edición. 2013 (vid. pág. 22).

[35] VisualParadim. *Software Design Tools for Agile Teams, with UML, BPMN and More*. 2017. URL: `https://www.visual-paradigm.com/` (vid. pág. 23).

[36] Ricardo Yañez-Valdeza et al., «Diseño de un sistema leva seguidor cilíndrico desmodrómico para generar desplazamientos lineales reciprocantes». En: () (vid. pág. 8).

Generado con LaTeX: 9 de julio de 2018: 7:43am

Apéndices

APÉNDICE A

Anexos

A.1. Metodología AUP variante UCI

Figura A.1. Fases de la metodología AUP variante UCI.

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración Construcción Transición	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes el proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Figura A.2. Disciplinas de la metodología AUP variante UCI.

Disciplinas AUP	Disciplinas Variación AUP-UCI	Objetivos Disciplinas(Variación AUP-UCI)
Modelado	Modelado de negocio	El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes: 1- Casos de Uso del Negocio (CUN). 2- Descripción de Proceso de Negocio (DPN). 3- Modelo Conceptual (MC). A partir de las variantes anteriores se condicionan cuatro escenarios para modelar el sistema en la disciplina Requisitos.
	Requisitos	El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos [Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP)], agrupados en cuatro escenarios condicionados por el Modelado de negocio. Ver 4.2.1 Escenarios para la disciplina Requisitos.
	Análisis y diseño	En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.
Implementación	**Implementación**	**En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.**
Prueba	Prueba interna	En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas.
	Prueba de liberación	Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
	Prueba de aceptación	Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.
Gestión de configuración	Se cubren con el área de procesos PP PMC y CM que propone CMMI-DEV v1.3. Las mismas son áreas de procesos de gestión y soporte respectivamente.	Consultar en mejoras.prod.uci.cu los libros de procesos de cada una de estas áreas.
Gestión de Proyecto		
Entorno		

Figura A.3. Roles de la metodología AUP variante UCI.

Roles AUP	Roles Variación AUP-UCI	Responsabilidades Roles (Variación AUP-UCI)
Administrador de proyecto	Jefe de proyecto	Las habilidades y competencias de cada uno de los roles definidos para la Variación de AUP-UCI se pueden consultar en mejoras.prod.uci.cu
	Planificador	
Ingeniero de procesos	Analista Arquitecto de información	
Modelador ágil	(Opcional)	
Desarrollador	Desarrollador	
Administrador de la configuración	Administrador de la Configuración	
Stakeholder	Stakeholder (Cliente/Proveedor de requisitos)	
Administrador de pruebas	Administrador de calidad	
Probador	Probador	
Administrador de BD	Arquitecto de software (Sistema)	
	Administrador de BD	

A.2. Historias de Usuario

Tabla A.1. Historia de usuario #2

Historia de usuario	
Número: 2	**Nombre:** Insertar los parámetros generales según el tipo de mecanismos
Programador: Reydel Baños Acosta	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 4 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 2 horas

Continúa en la próxima página

Tabla A.1. Continuación de la página anterior

Descripción: 1- Objetivo: Permitir la inserción de los parámetros generales del mecanismo de leva.

2- Acciones para lograr el objetivo (precondiciones y datos): Para insertar parámetros generales del mecanismo de levas hay que:

- Tener en cuenta los siguientes datos de entrada: función de desplazamiento,tramo actual y radio del seguidor y ancho del seguidor así como la excentricidad del seguidor con respecto a la leva.
- El radio y ancho del seguidor deben ser del tipo de dato "*double*".
- El ángulo de presión debe ser del tipo de dato "*double*" cuando se introduce por la interfaz, luego para realizar los demás cálculos hay que convertirlo a grado, esto generalmente se hace multiplicando su valor por $\pi \div 180.0$

3- Flujo de la acción a realizar:

- El usuario introducirá correctamente los datos necesarios y si selecciona la opción "*Calculate*" y estos tienen errores, se le debe advertir para que pueda rectificarlos.
- Si se selecciona el botón "*Cancel*" se cierra la ventana.

Observaciones: Permitir la selección del mecanismo de leva .

Prototipo elemental de interfaz gráfica de usuario

Tabla A.2. Historia de usuario #3

Historia de usuario	
Número: 3	**Nombre:** Insertar los parámetros específicos del mecanismo de leva de tipo disco
Programador: Reydel Baños Acosta	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 hora
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 0.5 horas

Continúa en la próxima página

Tabla A.2. Continuación de la página anterior

Descripción:

1- Objetivo: Permitir la inserción de los parámetros específicos del mecanismo de leva de tipo disco.

2- Acciones para lograr el objetivo (precondiciones y datos): Para insertar parámetros del mecanismo hay que:

- Tener en cuenta en cuantos tramos será construido el mecanismo.
- El radio base debe ser del tipo de dato "*double*" y tiene que ser mayor igual que cero.

3- Flujo de la acción a realizar:

- El usuario introducirá correctamente los datos necesarios y si selecciona la opción "*Calculate*" y estos tienen errores, se le debe advertir para que pueda rectificarlos.
- Si se selecciona el botón "*Cancel*" se cierra la ventana.

Observaciones: Permitir la selección del mecanismo de leva .

Prototipo elemental de interfaz gráfica de usuario

Tabla A.3. Historia de usuario #4

Historia de usuario	
Número: 4	**Nombre:** Insertar los parámetros específicos del mecanismo de leva de tipo lineal
Programador: Reydel Baños Acosta	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 hora
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 0.5 horas

Continúa en la próxima página

Tabla A.3. Continuación de la página anterior

Descripción:
1- Objetivo: Permitir la inserción de los parámetros específicos del mecanismo de leva de tipo lineal.

2- Acciones para lograr el objetivo (precondiciones y datos): Para insertar parámetros del mecanismo de leva hay que:

- Tener en cuenta en cuantos tramos será construido el mecanismo.
- El tamaño de la leva debe ser del tipo de dato "*double*".

3- Flujo de la acción a realizar:

- El usuario introducirá correctamente los datos necesarios y si selecciona la opción "*Calculate*" y estos tienen errores, se le debe advertir para que pueda rectificarlos.
- Si se selecciona el botón "*Cancel*" se cierra la ventana.

Observaciones: Permitir la selección del mecanismo de leva .

Prototipo elemental de interfaz gráfica de usuario

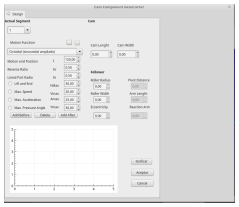


Tabla A.4. Historia de usuario #4

Historia de usuario	
Número: 5	**Nombre:** Insertar los parámetros específicos del mecanismo de leva de tipo cilíndrico
Programador: Reydel Baños Acosta	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 hora
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 0.5 horas

Continúa en la próxima página

Tabla A.4. Continuación de la página anterior

Descripción:

1- Objetivo: Permitir la inserción de los parámetros específicos del mecanismo de leva de tipo cilíndrico.

2- Acciones para lograr el objetivo (precondiciones y datos): Para insertar parámetros del mecanismo hay que:

- Tener en cuenta en cuantos tramos será construido el mecanismo.
- El radio base del cilindro debe ser del tipo de dato "*double*" y tiene que ser mayor igual que cero.

3- Flujo de la acción a realizar:

- El usuario introducirá correctamente los datos necesarios y si selecciona la opción "*Calculate*" y estos tienen errores, se le debe advertir para que pueda rectificarlos.
- Si se selecciona el botón "*Cancel*" se cierra la ventana.

Observaciones: Permitir la selección del mecanismo de leva .

Prototipo elemental de interfaz gráfica de usuario

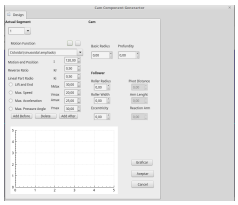


Tabla A.5. Historia de usuario #5

Historia de usuario	
Número: 6	**Nombre:** Mostrar el gráfico de desplazamiento del resultado de los cálculos
Programador: Reydel Baños Acosta	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 3 horas
Riesgo en Desarrollo: Alta	**Tiempo Real:** 4 horas

Continúa en la próxima página

Tabla A.5. Continuación de la página anterior

Descripción: **1- Objetivo:** Permitir que se muestre el gráfico de desplazamiento. newline **2- Acciones para lograr el objetivo (precondiciones y datos):** Para que se muestren el gráfico de cálculos de las propiedades de los mecanismos de leva hay que: • Tener en cuenta los siguientes datos de entrada: radios base, función por tramos , final del movimiento y ángulo de presión. • Estos valores que se le introducen tienen que ser del tipo de dato " *double*". **3- Flujo de la acción a realizar:** • El usuario introducirá correctamente los datos necesarios y seleccionara la opción "*Calculate*", si tienen errores se le debe advertir al usuario para que pueda modificarlos". • Si se selecciona el botón "*Cancel*" se cierra la ventana.
Observaciones: Permitir la selección del mecanismo de leva .
Prototipo elemental de interfaz gráfica de usuario 

Tabla A.6. Historia de usuario #7

Historia de usuario	
Número: 7	**Nombre:** Realizar modelado del mecanismo de leva
Programador: Reydel Baños Acosta	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 420 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 300 horas
Descripción: **1- Objetivo:** Permitir realizar el modelado del mecanismo de leva **2- Acciones para lograr el objetivo (precondiciones y datos):** Para realizar el modelado del mecanismo de leva hay que: • Tener los datos introducido en la ventana principal sin errores. **3- Flujo de la acción a realizar:** • El usuario debe seleccionar el "*DiscCam*" para poder acceder a la función que crea la el mecanismo de leva. • Selecciona el botón "Calculate" para que los datos introducidos en la interfaz, permitan que se muestre el gráfico de la función. • Seleccionar el botón "aceptar" se cierra la ventana principal y en el visor se muestra el mecanismo de leva.	

Continúa en la próxima página

Tabla A.6. Continuación de la página anterior

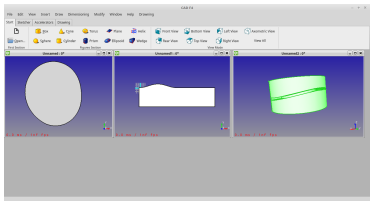
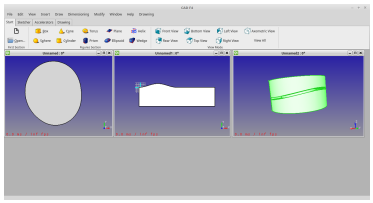
Observaciones: Permitir la selección del mecanismo de leva .
Prototipo elemental de interfaz gráfica de usuario 

Tabla A.7. Historia de usuario #8

Historia de usuario	
Número: 8	**Nombre:** Modelar el mecanismo de leva
Programador: Reydel Baños Acosta	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 20 horas
Riesgo en Desarrollo: Alta	**Tiempo Real:** 15 horas
Descripción: 1- Objetivo: Permitir visualizar el mecanismo de leva en tres dimensiones **2- Acciones para lograr el objetivo (precondiciones y datos):** Para Visualizar el mecanismo de levas en tres dimensiones hay que: • Tener los datos introducido en la ventana principal sin errores. **3- Flujo de la acción a realizar:** • El usuario debe seleccionar en ‘ la opción‘*aceptar*” para poder acceder a las funciones que crean el mecanismo de leva. • Selecciona el botón “Calculate” para que los datos introducidos en la interfaz, permitan que se muestre el gráfico de la función. • Seleccionar el botón “Aceptar” se cierra la ventana principal y en el visor se muestra el mecanismo de leva	
Observaciones: Permitir la selección del mecanismo de leva .	
Prototipo elemental de interfaz gráfica de usuario 	

APÉNDICE B

Tablas de casos de pruebas

Tabla B.1. Diseño de Casos de Prueba de la Historia de Usuario " Seleccionar mecanismo de leva según su tipo".

Descripción general			
Permitir seleccionar mecanismo de leva según su tipo.			
SC 1 Seleccionar mecanismo de leva según su tipo			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*Cam Disc*".	Selecciona el "*ComboBox*" "*Cam Disc*".	Brinda la posibilidad de seleccionar el mecanismo "*Cam Disc*".	Accelerators/ CamShaft /CamShaft.
EC 1.2 Opción seleccionar "*Cam Linear*".	Selecciona el "*ComboBox*" "*Cam Linear*".	Brinda la posibilidad de seleccionar el mecanismo "*Cam linear*".	Accelerators/ CamShaft /CamLinear.
EC 1.3 Opción seleccionar "*Cam Cylinder*".	Selecciona el "*Combobox*" "*Cam Cylinder*".	Brinda la posibilidad de seleccionar el mecanismo *Cam Cylinder*,	Accelerators/ CamShaft /CamCylinder.

Tabla B.2. Diseño de Casos de Prueba de la Historia de Usuario "Insertar los parámetros generales según el tipo de mecanismo".

Descripción general			
Permitir insertar parámetros generales según su el tipo mecanismo.			
SC 2 Permitir insertar parámetros generales según su el tipo mecanismo.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**

EC 1.1 Opción seleccionar "*Cam Disc*".	Selecciona el "*ComboBox*" "*Función por tramo*".	Brinda la posibilidad de seleccionar el tipo de función por tramo de los elementos (Cam Dis, Cam Linear,Cam Cylinder).	Accelerators/ CamShaft /Tramos.
EC 1.2 Opción seleccionar "*Final del movimiento*".	Selecciona el "*DobleSpinBox*" "*Final del movimiento*".	Brinda la posibilidad de seleccionar el valor de la (Elongación del seguidor).	Accelerators/ CamShaft /Tramos.

Tabla B.3. Insertar los parámetros específicos del mecanismo de leva de tipo disco.

Descripción general			
Permitir insertar los parámetros específicos del mecanismo de leva de tipo disco .			
SC 3 Insertar los parámetros específicos del mecanismo de leva de tipo disco			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*Cam Disc*".	Selecciona el "*DobleSpinBox*" "*Radio base*".	Brinda la posibilidad de seleccionar del mecanismo de la "*Radio base*".	Accelerators/ CamShaft /CamShaft.
EC 1.2 Opción seleccionar "*Cam wight*".	Selecciona el "*DobleSpinBox*" "*Cam wight*".	Brinda la posibilidad de seleccionar el valor de la "*Cam Wight*".	Accelerators/ CamShaft /CamShaft.
EC 1.3 Opción seleccionar "*Excentricidad*".	Selecciona el "*DobleSpinBox*" "*Excentricidad*".	Brinda la posibilidad de seleccionar el valor de la "*Excentricidad*".	Accelerators/ CamShaft /CamShaft.

Tabla B.4. Diseño de Casos de Prueba de la Historia de Usuario " Insertar los parámetros específicos del mecanismo de leva de tipo lineal".

Descripción general			
Insertar los parámetros específicos del mecanismo de leva de lineal. lineal.			
SC 4 Insertar los parámetros específicos del mecanismo de leva de tipo lineal tipo			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*CamLinear* ".	Selecciona el "*DobleSpinBox*" "*Cam Lenght*".	Brinda la posibilidad de seleccionar del mecanismo la propiedad "*Cam lenght*".	Accelerators/ CamShaft /CamLinear.

EC 1.2 Opción seleccionar "*CamLinear*".	Selecciona el "*DobleSpinBox*" "*Cam wight*".	Brinda la posibilidad de seleccionar el valor de la "*Pressure Angle*".	Accelerators/ CamShaft /CamLinear.
EC 1.3 Opción seleccionar "*CamLinear*".	Selecciona el "*DobleSpinBox*" "*Excentricidad*".	Brinda la posibilidad de seleccionar el valor de la "*Excentricidad*".	Accelerators/ CamShaft /CamLinear.

Tabla B.5. Diseño de Casos de Prueba de la Historia de Usuario " Insertar los parámetros específicos del mecanismo de leva de tipo cilíndrico".

Descripción general			
Permitir Insertar los parámetros específicos del mecanismo de leva de tipo cilíndrico.			
SC 5 Insertar los parámetros específicos del mecanismo de leva de tipo cilíndrico			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*Profundidad de la ranura*".	Selecciona el "*DobleSpinBox*" "*Profundidad*".	Brinda la posibilidad de seleccionar la propiedad del mecanismo de "*Profundidad*".	Accelerators/ CamShaft /CamCylinder.
EC 1.2 Opción seleccionar "*Radio Base*".	Selecciona el "*DobleSpinBox*" "*Radio Base*".	Brinda la posibilidad de seleccionar el valor del (Radio Base).	Accelerators/ CamShaft /CamCylinder.

APÉNDICE C

Tablas de casos de pruebas

Tabla C.1. Diseño de Casos de Prueba de la Historia de Usuario “Mostrar el gráfico de desplazamiento del resultado de los cálculos de los mecanismos de leva.”.

Descripción general			
Permitir mostrar el gráfico de desplazamiento del resultado de los cálculos de los mecanismos de leva.			
SC 6 Mostrar el gráfico de desplazamiento del resultado de los cálculos de los mecanismos de leva			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar (Cam Disc,Cam Linear,Cam Cylinder).	Selecciona el combox de la ventana principal y se muestra en la parte superior de la ventana la opción de seleccionar el mecanismo.	Muestra una gráfico en la parte inferior izquierda de la interfaz de diseño.	Accelerators / CamShaft /CamShaft.
EC 1.2 Opción seleccionar “*Radio Base*”.	Selecciona el “*DobleSpinBox*” “*Radio Base*”.	Brinda la posibilidad de seleccionar el valor del (Radio Base).	Accelerators / CamShaft /CamShaft.
Opción seleccionar “*función de desplazamiento*”.	Selecciona el “*Combobox*” (motion function).	Brinda la posibilidad de seleccionar el valor de la (función de movimiento)	Accelerators / CamShaft /CamShaft.

Tabla C.2. Diseño de Casos de Prueba de la Historia de Usuario “Mostrar los errores al diseñar el componente”.

Descripción general
Permitir Mostrar los errores al diseñar el componente tipo.

SC 7 Mostrar los errores al diseñar el componente tipo.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
Opción seleccionar "*Cam Disc*".	Selecciona el "*Botton*" "*calculate*".	Brinda la posibilidad de seleccionar el mecanismo de la "*Cam Disc*".	Accelerators/ CamShaft /CamShaft.
EC 1.2 Opción seleccionar "*Radio Base*".	Selecciona el "*DobleSpinBox*" "*Radio Base*".	Brinda la posibilidad de seleccionar el valor de la "*Radio Base*".	Accelerators/ CamShaft /CamShaft.
EC 1.3 Opción seleccionar "*Disc Cam*".	Selecciona el "*Combobox*" (Disc Cam,Disc Linear, Disc cylinder)	Accelerators/ CamShaft /CamShaft.	Accelerators/ CamShaft /CamShaft.

Tabla C.3. Diseño de Casos de Prueba de la Historia de Usuario " Realizar modelado del mecanismo".

Descripción general			
Permitir el modelado del mecanismo			
SC 8 Realizar modelado del mecanismo .			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*aceptar*".	Selecciona el "*botton*" "*aceptar*".	Brinda la posibilidad de seleccionar el tipo de pieza que va ser visualizada en este caso el mecanismo de leva .	Accelerators/ CamShaft/ Camshaft.
EC 1.2 Insertar los parámetros necesarios para la construcción.	Introducir parámetros como "*función de desplazamiento*", "*angulo de presíon*", "*Excentricidad*", "*Linear Ratio*" y "*Radio Reverse*".	Brinda la posibilidad de introducir los parámetros necesarios para su construcción.	Accelerators/ CamShaft/ Tramos.
EC 1.3 Visualizar el mecanismo.	Selecciona el botón"*aceptar*".	Brinda la posibilidad de visualizar el modelo en 3D en el visor.	Accelerators/ Cam Shaft/ Camshaft