Universidad de las Ciencias Informáticas

Facultad 4

Funcionalidades para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas

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

Autor:

Manuel Fornés Martínez

Tutores:

Dr.C Augusto Cesar Rodríguez Medina

Ing. Gustavo García González

Declaración de autoría

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

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2017.

_____

Manuel Fornés Martínez

Firma del Autor

_____ _____

Dr.C Augusto Cesar Rodríguez Medina

Firma del Tutor

Ing. Gustavo García González

Firma del Tutor

La juventud tiene que crear. Una juventud que no crea es una anomalía realmente (…)

Ernesto "Che" Guevara de la Serna

Dedicatoria:

Esta tesis está dedicada, a mi madre por ser la persona que me ha acompañado durante todo mi trayecto estudiantil. A mi padre quien con sus consejos ha sabido guiarme. A mi hermana gemela por estar siempre a mi lado. Y por último pero no menos importante a mi abuelo Manuel de Jesús Fornés Bonne quien estuvo conmigo en aquellos momentos en que la vida fue más difícil para mí.

Agradecimientos:

Agradecer a mi papá que a pesar de su fuerte carácter siempre ha estado en los momentos que lo he necesitado. A mi mamá por ser la mujer que siempre ha cumplido todos mis caprichos. A mi hermana gemela por ser la parte de mi cuerpo que más quiero. Ellos han sido la familia ideal en el transcurso de estos años sin pedir nada a cambio.

Agradecer a los amigos que son la familia que elegimos, a Ernesto, Elí, Josue y Raciel por estar ahí desde los primeros años de la mini-uci de Granma hasta este momento.

También a los amigos que me acogieron en la llegada a la uci, a Javier, Jeane, Oscar, Adrian.

Agradecer a Joel por ser aquel maestro que me ha sabido inculcar valores deportivos y aquel aire de grandeza que siempre tiene. También agradecer a los amigos del Kenpo, Sergio y al equipo del FITT-CUBA.

Agradecer a mis amigos y compañeros de proyecto, Sandy, Gustavo, Julio, Armando y el profesor Augusto.

Y por último pero no menos importante agradecerle a mi abuelo Manuel de Jesús Fornés Bonne por ser aquel hombre que dejo de ser ese abuelo para convertirse en aquel padre autentico, es a él que está dedicada esta tesis.

Resumen

El presente trabajo contiene los resultados generales de un proceso de investigación y desarrollo, cuyo objetivo fundamental, es desarrollar funcionalidades con base en herramientas de código abierto disponibles, para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas, destinadas a una aplicación informática de diseño asistido por computadora. Las decisiones que definieron el resultado estuvieron fundamentadas en una investigación previa, en la que se recopiló y procesó información acerca de sistemas comerciales y libres, con funcionalidades para el modelado de superficies con los requerimientos mencionados, el estudio de los conceptos para la creación de curvas y superficies, así como las interfaces de programación de aplicaciones, término que es más conocido por la denominación en inglés *Application Programing Inteface* (API), existentes en la tecnología Open Cascade para el modelado de las mismas. Las funcionalidades para el modelado de superficies interpoladas desarrolladas en el presente trabajo son: modelado de superficies Bézier, B-Spline y NURBS, suavizado de plano y barrido de curvas B-Spline.

Palabras claves: diseño asistido por computadora, modelado, perfiles curvos, superficies interpoladas.

Índice

Introducción

El presente trabajo de investigación y desarrollo posee sus bases en el proyecto Soluciones Informáticas para la Ingeniería y la Industria (SIPII) perteneciente al Departamento de Ciencias Básicas de la Facultad 4, el cual tiene como propósito desarrollar una aplicación para el Diseño Asistido por Computadoras (CAD, por sus siglas en inglés), destinada a los sectores de la industria nacional vinculados a la actividad de diseño. En este contexto una de las tareas fue desarrollar funcionalidades para el proceso de modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas Bézier, B-Spline y NURBS.

La idea del mencionado desarrollo surgió de la necesidad de funcionalidades para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas, destinadas a un sistema informático que permita resolver algunos de los problemas que enfrenta la industria nacional en el área de las tecnologías para el diseño asistido por computadora, asociados a las restricciones del bloqueo económico y comercial impuesto por el gobierno de los Estados Unidos de América contra Cuba.

Con el objetivo de eliminar las restricciones impuestas al país y teniendo en cuenta la política actual de soberanía tecnológica, se emplearon tecnologías de código abierto como la versión comunitaria de Open CASCADE, el marco de trabajo Qt y el código fuente de la aplicación FreeCAD y Salome-Meca.

El documento está estructurado en tres capítulos; en el primero se exponen los aspectos generales de la fundamentación teórica del trabajo, iniciando con la conformación del perfil de la investigación; se incluye un estudio de sistemas CAD, las propiedades y características de las curvas y superficies Bézier, B-Spline y NURBS, así como la metodología, herramientas y tecnologías para el desarrollo. En el segundo capítulo se muestra la propuesta de solución, se realiza el levantamiento de requisitos, el modelo del dominio, la descripción de las funcionalidades, la arquitectura, los patrones de diseño, el diagrama de secuencia, el modelo conceptual y las historias de usuarios. En el tercero se presenta elementos correspondientes a la implementación, como: los estándares de codificación, diagrama de componente y las pruebas realizadas a las funcionalidades.

1 Fundamentación Teórica

En este capítulo se exponen aspectos iniciales sobre la situación problemática que da origen al proceso de investigación. Además, recoge información esencial de sistemas homólogos, curvas y superficies Bézier, NURBS y B-Spline. También, aborda temas como las metodologías y las herramientas de desarrollo a emplear.

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

En la actualidad hay una amplia gama de programas CAD propietarios, como AutoCAD, Inventor, Catia, SolidEdge, SolidWorks, Rhinoceros entre otros. Todas estas herramientas ofrecen soluciones específicas a segmentos verticales de la industria (ingeniería civil, mecánica, industrial etc.)

Los sistemas antes mencionados poseen módulos para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas, estas representan con precisión objetos geométricos tales como líneas, círculos, elipses, esferas y toroides, así como formas geométricas libres como superficies topográficas, la configuración del cuerpo humano y otras superficies irregulares, las cuales son esenciales para el diseño de piezas tipo chapa en la industria aeronáutica, aeroespacial, donde se requiere el desarrollo de superficies complejas; también son ampliamente empleados en la industria naval y automotriz que emplean estos tipos de perfiles en piezas tipo chapa.

El gran problema de los sistemas comerciales, en el caso concreto de nuestro país, es que por las restricciones del bloqueo impuesto por el gobierno de los Estados Unidos, no es posible acceder a los mismos; no obstante, aún sin la existencia del bloqueo, los costos de las licencias bajo las cuales se distribuyen son elevados.

Una exigencia de proyecto vinculado al grupo de investigación “Soluciones Informáticas para la Ingeniería y la Industria”, destinado a crear alternativas propias al uso de los sistemas propietarios (que por demás hay que “crackear” para poder utilizarlos y por tanto no pueden tener una aprobación institucional) ha sido el desarrollo de funcionalidades para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas.

Entre las herramientas CAD de código abierto que podría emplear un ingeniero para su uso, con el fin de eliminar las dificultades antes mencionadas, están el LibreCAD y FreeCAD, pero estas no cuentan con funcionalidades para el modelado de curvas y superficies interpoladas que poseen sus similares propietarios, por lo que el modelado de este tipo de piezas se realiza con un mayor grado de dificultad y con un elevado tiempo de trabajo. En el ámbito de las herramientas libres de Ingeniería Asistida por Computadora (CAE, por sus siglas en inglés) existe la aplicación “Salome-Meca”, la cual posee dos

funcionalidades para el modelado de superficies en uno de sus módulos (“Geometry”), insuficientes al ser comparadas con la de los sistemas propietarios.

Partiendo de este análisis se fórmula el siguiente **problema de investigación:**

¿Cómo modelar prototipos virtuales de piezas tipo chapa con perfiles de curvas y superficies interpoladas?

El **objeto de estudio** de la investigación se centra en las funcionalidades para el modelado de piezas tipo chapa, teniendo como **campo de acción**, las funcionalidades para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas.

Para solucionar el problema planteado se propone como **objetivo general**:

Desarrollar funcionalidades para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas, basado en funcionalidades disponibles en aplicaciones de código abierto como FreeCAD y el módulo “Geometry” de la aplicación “Salome-Meca”.

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

- Diseñar las funcionalidades para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas.
- Obtener funcionalidades para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas.
- Realizar pruebas a las funcionalidades desarrolladas.

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

- Asimilación de los conceptos y tecnologías requeridas para el desarrollo de las funcionalidades.
- Definición del perfil y diseño de la investigación.
- La asimilación de los fundamentos teóricos sobre curvas y superficies Bézier, NURBS y B-Spline.
- Búsqueda de las API existentes en la tecnología Open Cascade para el trabajo con las mismas.
- Estudios 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 existentes en sistemas propietarios, así como consideraciones de los potenciales usuarios.
- Definición de la arquitectura.

- Definición los patrones de diseño.
- Determinación de la estructura de clases.
- Modelado del flujo de datos.
- Diseño de las interfaces gráficas de las funcionalidades.
- Implementación de funcionalidades destinadas al modelado de piezas tipo chapa con perfiles curvos, superficies NURBS y B-Spline.
- Comprobación de las funcionalidades implementadas (pruebas unitarias y funcionales).

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

Métodos teóricos:

- Análisis y síntesis: se empleó para la construcción y desarrollo de la teoría, profundización en el tema y la sistematización del conocimiento.
- Modelado: se empleó en la generación de los artefactos que permitan lograr una mejor comprensión entre el equipo de desarrollo y las personas relacionadas.

Métodos empíricos:

- Observación: se empleó para caracterizar las soluciones, teniendo en cuenta distintos datos de otras aplicaciones y así establecer los requisitos con las principales funcionalidades de estos.

1.2 Curvas y Superficies.

Se describe conceptos básicos de las curvas y superficies Bezier, B-Spline y NURBS para un mejor entendimiento al uso de las APIs que brinda Open Cascade.

1.2.1 Curva de Bézier.

Una curva de Bézier se puede representar como racional o no racional.

- ✔ Una curva no racional de Bézier se define por una tabla de polos (también llamados puntos de control).
- ✔ Una curva racional de Bézier se define por una tabla de polos con pesos variables.

Estos datos son manipulados por dos matrices paralelas: Una tabla de polos, que es una matriz de puntos y la tabla de pesos, que es una matriz de reales. Los límites de estas matrices son 1 y el número de polos de la curva. Los polos de la curva son "puntos de control" utilizados para deformar la curva (Center, Rarouki y Rajan 1987). El primer polo es el punto de inicio de la curva y el último polo es

el punto final de la curva. El segmento que une el primer polo al segundo polo es tangente a la curva en su punto de inicio y el segmento que une el último polo con el polo del segundo de la última es la tangente a la curva en su punto final (O'Rourke 1998).

Si los pesos de todos los polos son iguales, la curva es polinómica; por lo tanto, es una curva no racional. La curva no racional es un caso especial y de uso frecuente, los pesos se definen y utilizan solo en el caso de una curva racional (Struik 2012). El grado de una curva de Bézier es igual al número de polos menos 1, este debe ser mayor o igual que 1; si el primer y el último punto de control de la curva de Bézier son el mismo punto entonces la curva es cerrada. No es posible construir una curva de Bézier con pesos negativos (Cormen et al. 2001).

1.2.2 Curva de B-Spline.

Una curva B-Spline puede ser (Prautzsch, Boehm y Paluszny 2013):

- ✔ Uniforme o no uniforme.
- ✔ Racional o no racional.
- ✔ Periódico o no periódico.

Una curva B-Spline se define por (Prautzsch, Boehm y Paluszny 2013):

- ✔ Grado.
- ✔ Carácter periódico o no periódico.
- ✔ Una tabla de polos (también llamada puntos de control), con los pesos asociados si la curva B-Spline es racional.

Los polos de la curva son "puntos de control" utilizados para deformar la curva. Si la curva no es periódica, el primer polo es el punto inicial de la curva y el último polo es el punto final de la curva (Agoston 2005). El segmento que une el primer polo al segundo polo es la tangente a la curva en su punto inicial y el segmento que une el último polo con el polo del segundo desde el último es tangente a la curva en su punto final, si la curva es periódica estas propiedades geométricas no se verifican; si los pesos de todos los polos son iguales la curva tiene una ecuación polinómica, por lo tanto, es una curva no racional (Prautzsch, Boehm y Paluszny 2013).

Para una curva B-Spline, la tabla de nudos es una secuencia creciente de reales sin repetición donde las multiplicidades definen la repetición de los nudos (Goldman y Lyche 1993). La multiplicidad Mult (i) del nudo Knot (i) de la curva BSpline se relaciona con el grado de continuidad de la curva en el nudo Knot (i), que es igual a Degree - Mult (i) donde Degree es el grado de la curva B-Spline (Prautzsch, Boehm y Paluszny 2013).

Si los nudos están regularmente espaciados (es decir, la diferencia entre dos nudos consecutivos, es constante), se pueden identificar tres casos específicos y frecuentemente usados de distribución de nudos (Prautzsch, Boehm y Paluszny 2013):

- ✔ Uniforme - si todas las multiplicidades son iguales a 1.
- ✔ Casi Uniforme - si todas las multiplicidades son iguales a 1, excepto el primero y el último nudo que tienen una multiplicidad de Grado + 1, donde Grado es el grado de la curva B-Spline.
- ✔ Piecewise Bézier - si todas las multiplicidades son iguales a Grado excepto el primer y último nudo que tienen una multiplicidad de Grado + 1, donde Grado es el grado de la curva de B-Spline.

Si la curva B-Spline no es periódica: Los límites de las tablas de polos y pesos son 1 y NbPoles, donde NbPoles es el número de polos de la curva B-Spline. Los límites de las tablas Nudos y Multiplicidades son 1 y NbKnots, donde NbKnots es el número de nudos de la curva B-Spline (Prautzsch, Boehm y Paluszny 2013).

1.2.3 Curva NURBS

La geometría NURBS tiene cinco cualidades esenciales que la convierten en la opción ideal para el modelado asistido por ordenador. Una curva NURBS se define mediante tres elementos: grados, puntos de control y nodos (Rogers 2001).

El grado es un número entero positivo, este número es 1, 2, 3 o 5, pero puede ser cualquier número entero positivo. Las líneas y polilíneas NURBS son normalmente de grado 1, los círculos son de grado 2 y la mayoría de las formas libres son grado 3 o 5; a veces se utilizan los siguientes términos: lineal, cuadrático, cúbico y quintico, lineal significa de grado 1, cuadrático significa de grado 2, cúbico significa de grado 3 y quintico significa grado 5 (Piegl y Tiller 2012).

Es posible que vea referencias del orden de una curva NURBS, este es un número entero positivo igual a (grado+1). En consecuencia, el grado es igual a (orden-1), existe la posibilidad de incrementar los grados de una curva NURBS sin cambiar su forma. Pero no es posible reducir el grado de una curva NURBS y no cambiar su forma (Preparata y Shamos 2012).

Los puntos de control son una lista de puntos de grado+1 como mínimo. Una de las maneras más sencillas de cambiar la forma de una curva NURBS es mover los puntos de control; estos tienen un número asociado denominado peso, con algunas excepciones, los pesos son números positivos («Computacion Grafica: CURVAS NURBS» [sin fecha]). Cuando todos los puntos de control de una curva tienen el mismo peso (normalmente 1), la curva se denomina no racional. De lo contrario, la curva se denomina racional. En NURBS, la R significa racional e indica que una curva NURBS tiene la

posibilidad de ser racional. A la práctica, la mayoría de las curvas NURBS son no racionales. Algunas curvas, círculos y elipses NURBS, ejemplos significativos, son siempre racionales (Piegl y Tiller 2012).

Los nodos son una lista de números de grado es igual a N-1, donde N es el número de puntos de control. A veces esta lista de números se denomina vector nodal. En este contexto, la palabra vector no significa una dirección 3D.

Esta lista de números de nodos debe cumplir varias condiciones técnicas. El modo estándar para asegurar que las condiciones técnicas se cumplan es requerir que el número se mantenga igual o aumente a medida que vaya bajando en la lista y limitar el número de valores duplicados a que no sea superior al grado (Piegl y Tiller 2012).

Se dice que un valor nodal es un nodo de multiplicidad total si el grado se ha duplicado varias veces. Si una lista de nodos se inicia con un nodo de multiplicidad completa, la siguen nodos simples, termina con un nodo de multiplicidad completa y los valores se espacian uniformemente, entonces los nodos son uniformes. Los nodos que no son uniformes se denominan no uniformes. Las letras N y U de la palabra NURBS significan "no uniforme" e indican que se permite que los nodos de una curva NURBS sean no uniformes (Piegl y Tiller 2012).

Los valores duplicados del nodo en la mitad de la lista del nodo hacen que una curva de NURBS sea menos suave. En caso extremo, un nodo de completa multiplicidad en la mitad de la lista de nodos significa que hay un lugar en la curva NURBS que se puede doblar en un punto de torsión. Debido a que el número de nodos es igual a (N+grado-1), donde N es el número de puntos de control, si se agregan nodos también se agregan puntos de control, y si se quitan nodos se quitan puntos de control. Los nodos se pueden añadir sin cambiar la forma de la curva de NURBS. En general, quitar nodos cambiará la forma de una curva (Piegl y Tiller 2012).

Un error frecuente se produce cuando cada nodo se empareja con un punto de control, y ocurre sólo en las NURBS de grado 1 (polilíneas). Para curvas NURBS de grados más altos, existen grupos de nodos de 2 x grado que corresponden a grupos de puntos de control de grado (Piegl y Tiller 2012).

1.2.4 Superficie de Bézier

Describe una superficie de Bézier racional o no racional (Yamaguchi 2012).

- ✔ Una superficie no racional de Bézier se define por una tabla de polos (también conocidos como puntos de control).
- ✔ Una superficie racional de Bézier se define por una tabla de polos con diferentes pesos asociados.

Estos datos se manipulan utilizando dos matrices 2D asociativos (Yamaguchi 2012):

- ✔ La tabla de polos, que es una matriz 2D de puntos.
- ✔ La tabla de pesos, que es una matriz 2D de reales.

Los límites de estos matrices son:

- ✔ 1 y NbUPoles para los límites de fila, donde NbUPoles es el número de polos de la superficie en la dirección u paramétrica.
- ✔ 1 y NbVPoles para los límites de columna, donde NbVPoles es el número de polos de la superficie en la v dirección paramétrica

Los polos de la superficie, los "puntos de control", son los puntos utilizados para dar forma y remodelar la superficie. Comprenden una red rectangular de puntos: Los puntos (1, 1), (NbUPoles, 1), (1, NbVPoles) y (NbUPoles, NbVPoles) son los cuatro "vértices" paramétricos de la superficie. La primera columna de polos y la última columna de polos definen dos curvas de Bézier que delimitan la superficie en la v dirección paramétrica. Estas son las curvas isoparamétricas v correspondientes a los valores 0 y 1 del parámetro v (Chui 1988).

La primera fila de polos y la última fila de polos definen dos curvas de Bézier que delimitan la superficie en la dirección paramétrica “u”. Estas son las curvas isoparamétricas “u” correspondientes a los valores 0 y 1 del parámetro “u”. Si los pesos de todos los polos son iguales, la superficie tiene una ecuación polinómica, y por lo tanto es una "superficie no racional". La superficie no racional es un caso especial, pero de uso frecuente, donde todos los polos tienen pesos idénticos (Chui 1988).

Los pesos se definen y utilizan solo en el caso de una superficie racional. Esta característica racional se define en cada dirección paramétrica. Por lo tanto, una superficie puede ser racional en la dirección u paramétrica, y no racional en la v dirección paramétrica (Yamaguchi 2012).

Del mismo modo, el grado de una superficie se define en cada dirección paramétrica. El grado de una superficie de Bézier en una dirección paramétrica dada es igual al número de polos de la superficie en esa dirección paramétrica menos 1. Una superficie de Bézier también puede estar cerrada o abierta en cada dirección paramétrica. Si la primera fila de polos, es idéntica a la última fila de polos, la superficie se cierra en la dirección paramétrica u (Yamaguchi 2012).

Si la primera columna de polos, es idéntica a la última columna de polos, la superficie se cierra en la v dirección paramétrica. La continuidad de una superficie de Bézier es infinita en la dirección paramétrica u y en la dirección paramétrica (Yamaguchi 2012).

Superficie de B-Spline.

En cada dirección paramétrica, una superficie B-Spline puede ser (Chui 1988):

- ✔ Uniforme o no uniforme.

- ✔ Racionales o no racionales.
- ✔ Periódicos o no periódicos.

Una superficie B-Spline se define por (Chui 1988):

- ✔ Sus grados en las direcciones paramétricas u y v.
- ✔ Su característica periódica en las direcciones paramétricas u y v.
- ✔ Una tabla de polos, también llamados puntos de control (juntos con los pesos asociados si la superficie es racional).
- ✔ Una tabla de nudos, junto con las multiplicidades asociadas.

Los polos y los pesos se manipulan usando dos matrices asociativas 2D (Chui 1988):

- ✔ La tabla de polos, que es una matriz 2D de puntos.
- ✔ La tabla de pesos, que es una matriz 2D de reales.

Los límites de las matrices de polos y pesos son (Chui 1988):

- ✔ 1 y NbUPoles para los límites de fila (siempre que la superficie de B-Spline no sea periódica en la dirección u paramétrica), donde NbUPoles es el número de polos de la superficie en la dirección paramétrica u.
- ✔ 1 y NbVPoles para los límites de columna (siempre que la superficie B-Spline no sea periódica en la dirección v paramétrica), donde NbVPoles es el número de polos de la superficie en la v dirección paramétrica.

Los polos de la superficie son los puntos utilizados para dar forma y remodelar la superficie. Comprenden una red rectangular. Si la superficie no es periódica: Los puntos (1, 1), (NbUPoles, 1), (1, NbVPoles) y (NbUPoles, NbVPoles) son los cuatro "vértices" paramétricos de la superficie. La primera columna de polos y la última columna de polos definen dos curvas B-Spline que delimitan la superficie en la v dirección paramétrica. Estas son las curvas isoparamétricas que corresponden a los dos límites del parámetro v (Yamaguchi 2012).

La primera fila de polos y la última fila de polos definen dos curvas B-Spline que delimitan la superficie en la dirección u paramétrica. Estas son las curvas isoparamétricas u correspondientes a los dos límites del parámetro u. Si la superficie es periódica, estas propiedades geométricas no se verifican (Yamaguchi 2012).

Además, si los pesos de todos los polos son iguales, la superficie tiene una ecuación polinómica, y por lo tanto es una "superficie no racional". La superficie no racional es un caso especial, pero de uso frecuente, donde todos los polos tienen pesos idénticos (Yamaguchi 2012).

Los pesos se definen y utilizan solo en el caso de una superficie racional. La característica racional se define en cada dirección paramétrica. Una superficie puede ser racional en la dirección u paramétrica, y no racional en la dirección v paramétricav (Yamaguchi 2012).

Para una superficie B-Spline la tabla de nudos se compone de dos secuencias cada vez mayores de reales, sin repetición, una para cada dirección paramétrica. Las multiplicidades definen la repetición de los nudos (Yamaguchi 2012).

Una superficie B-Spline comprende múltiples parches contiguos, que son por sí mismos superficies polinómicas o racionales. Los nudos son los parámetros de las curvas isoparamétricas que limitan estos parches contiguos.

La multiplicidad de un nudo en una superficie B-Spline (en una dirección paramétrica dada) está relacionada con el grado de continuidad de la superficie en ese nudo en esa dirección paramétrica (Yamaguchi 2012):

- ✔ Grado de continuidad al nudo (i) = Grado - Multi (i) donde: Grado es el grado de la superficie B-Spline en la dirección paramétrica dada. Multi (i) es la multiplicidad del número de nudos i en la dirección paramétrica dada.

Hay algunos casos especiales, donde los nudos están regularmente espaciados en una dirección paramétrica (es decir, la diferencia entre dos nudos consecutivos es una constante) (Prautzsch, Boehm y Paluszny 2013).

- ✔ Uniforme - todas las multiplicidades son iguales a 1.
- ✔ Quasi-uniforme - todas las multiplicidades son iguales a 1, excepto el primer y último nudo en esta dirección paramétrica, y estas son iguales a Grado + 1.
- ✔ Piecewise Bézier - todas las multiplicidades son iguales a Grado excepto para el primer y último nudo, que son iguales a Grado + 1. Esta superficie es una concatenación de parches de Bézier en la dirección paramétrica dada.

Si la superficie B-Spline no es periódica en una dirección paramétrica dada, los límites de las tablas de nudos y multiplicidades son 1 y NbKnots, donde NbKnots es el número de nudos de la superficie B-Spline en esa dirección paramétrica (Yamaguchi 2012).

1.3 APIs existentes en Open Cascade para el modelado de curvas y superficies.

La tecnología Open Cascade ofrece diferentes clases y funcionalidades para la representación de curvas y superficies, un estudio de la documentación («Open CASCADE Technology» [sin fecha]) ha permitido identificarlas, las cuales son:

- **Para curvas:**
    1. Geom_BezierCurve (Crea una curva no racional de Bezier con un conjunto de puntos).
    1. Geom_BsplineCurve (Crea una curva no racional B-Spline con una lista de puntos, grado y nudos con sus multiplicidades).
    2. GeomAPI_PointsToBSpline (Aproxima una curva B-Spline dada una matriz de puntos).
    3. GeomAPI_Interpolate (Interpola una curva B-Spline dada una matriz de puntos).
    4. BrepBuilderAPI_NurbsConvert (Convierte una geometría en una NURBS).
- **Para superficies**:
    1. Geom_BezierSurface (Crea una superficie no racional con un conjunto de puntos).
    2. Geom_BsplineSurface (Crea una superficie no racional B-Spline con una lista de puntos, grado y nudos con sus multiplicidades).
    3. GeomAPI_PointsToBSplineSurface (Aproxima una superficie B-Spline dada una matriz de puntos).
- **Para aproximación de planos:**
    1. GeomPlate_BuildPlateSurface (Proporciona un algoritmo para construir una superficie que se ajusta a las restrinciones dadas de la curva o del punto).

1.4 Funcionalidades existentes para el modelado de curvas y superficies interpoladas

En la actualidad con el crecimiento de las ciencias informáticas, han surgido un grupo de sistemas con el objetivo de satisfacer las necesidades de los ingenieros y diseñadores mecánicos en su trabajo cotidiano, facilitándoles el diseño en múltiples planos de trabajoso entendimiento, reduciendo el tiempo de trabajo y aumentando la calidad del mismo. En este apartado se abordarán algunos de los sistemas propietarios y libres existentes en el mercado internacional.

1.4.1 Sistemas propietarios

1.4.1.1 Catia

En la barra de herramientas nos permite crear superficies para la generación del diseño a partir de elementos alámbricos.

La opción **extrude** permite la generación de un superficie a partir del barrido de una curva en una dirección dada con unos límites determinados (Cidoncha et al. 2007).

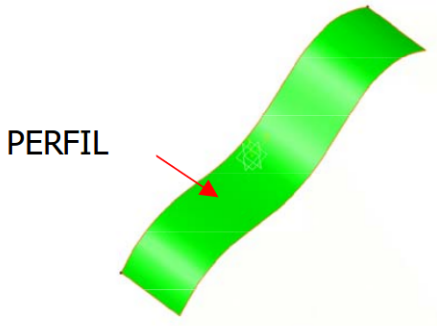


Fig 1 Extrude

La herramienta **revolve** genera una superficie de revolución a partir de una curva y el eje de giro (Cidoncha et al. 2007).

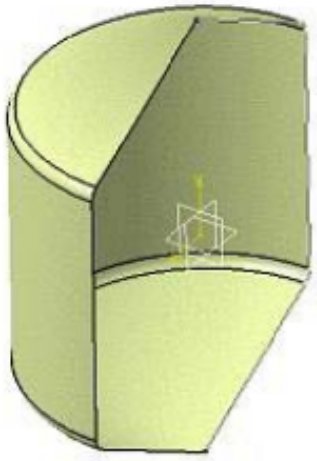


Fig 2 Revolve

La opción **Explicit** permite la generación de una superficie a partir del barrido de una curva por otras curva guias. Las secciones curvas están colocadas en planos perpendiculares a la espina seleccionada (Cidoncha et al. 2007).

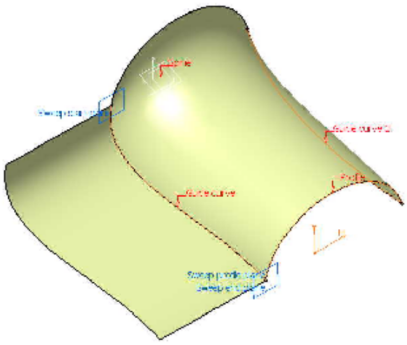


Fig 3 Explicit

La opción **Line** permite la generación de una superficie a partir del barrido de una recta por otra curva guía. Las secciones rectas están colocadas en planos perpendiculares a la espina seleccionada (Cidoncha et al. 2007).

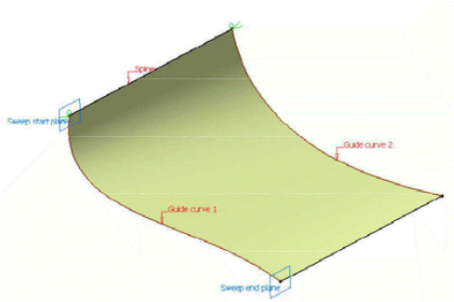


Fig 4 Line

La opción **Circle** permite la generación de una superficie a partir del barrido de un circulo por otra curva guía. Las secciones circulares deben estar colocadas en planos perpendiculares a la espina seleccionada (Cidoncha et al. 2007).

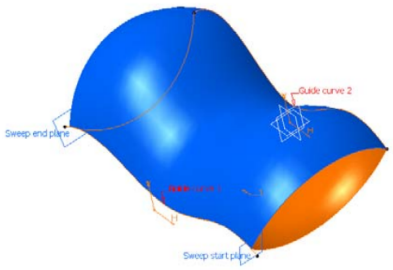


Fig 5 Circle

La herramienta de **multisection surfaces** permite la generación de superficies mediante el barrido de diferentes perfiles colocados perpendicularmente a la espina. Además permite que se le den una serie de curvas guías para conseguir que se adapte al diseño (Cidoncha et al. 2007).

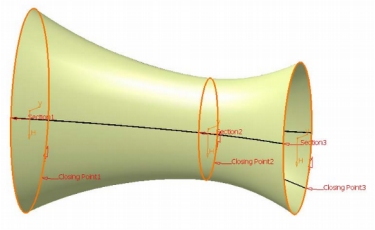


Fig 6 Mutisection

La herramienta de blend crea superficies de transición entre dos dadas. Para ello se seleccionan las curvas y el soporte al que pertenecen las curvas (Cidoncha et al. 2007).

1.4.1.2 Solid Edge

En términos de modelado de Solid Edge maneja una superficie con secciones transversales y curvas guía (Do Carmo 1990).

Superficie extruida: La única entrada necesaria es un boceto o perfil que contenga curvas o elementos analíticos (Do Carmo 1990).

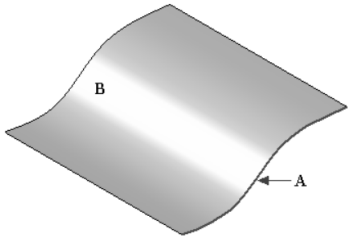


Fig 7 Extruida

Superficie por revolución: La única entrada necesaria es un boceto o perfil que contenga curvas o elementos analíticos y un eje de revolución (Do Carmo 1990).

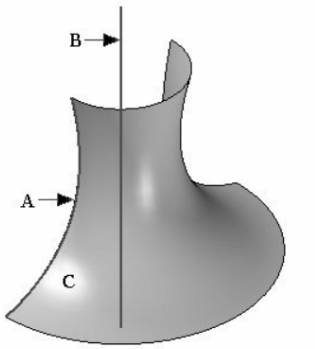


Fig 8 Revolución

El comando **Barrido** crea una superficie de construcción extruyendo un perfil a lo largo de una trayectoria (Do Carmo 1990).

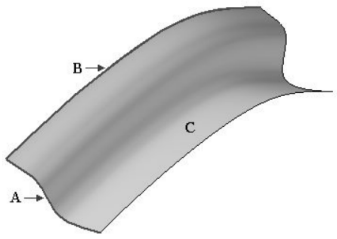


Fig 9 Barrido

Las **superficies por secciones** se construyen extruyendo dos o más secciones transversales para construir una operación (Do Carmo 1990).

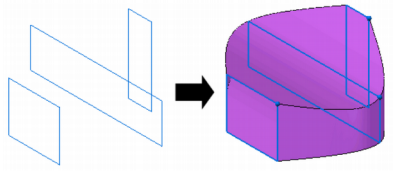


Fig 10 Secciones

El comando **superficie limitada** crea una superficie entre curvas o bordes. Este comando resulta útil cuando de sea llenar separaciones entre otras superficies adyacentes (Do Carmo 1990).

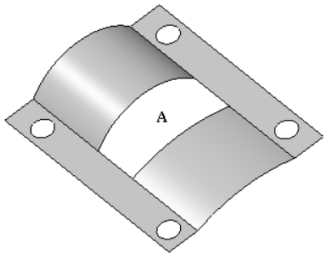


Fig 11 Limitada

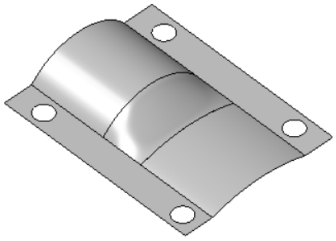


Fig 12 Limitada

BlueSurf es un comando de creación de superficies utilizado para generar superficies complejas pero altamente editables. Al igual que por **secciones** y **barrido**, **BlueSurf** utiliza secciones y curvas guía, y estas curvas antecesoras controlan el comportamiento de la superficie resultante.

En la pestaña **Estándar**, hay opciones para controlar la tangencia de secciones transversales y bordes de guía. El parámetro predeterminado para tangencia de sección transversal es Natural. El siguiente ejemplo muestra los resultados de un ajuste de control de tangencia natural (A) y perpendicular a sección (B) (Do Carmo 1990).

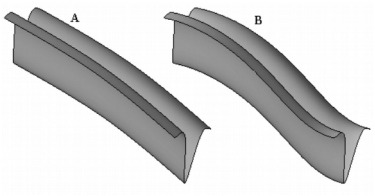


Fig 13 BlueSurf

1.4.1.3 Rhinoceros

La mayoría de las superficies que se crean en Rhino estarán basadas en una curva u otra superficie. Rhino contiene una gran variedad de construcciones de superficie, incluyendo superficies de forma libre que se adaptan a los puntos, construcciones de superficies equidistantes, con redondeos rodantes y superficies mezcladas.

Superficie a partir de Aristas: Puede crear una superficie a partir de tres o cuatro curvas que forman los lados de la superficie («Create surfaces | Rhino 3-D modeling» 2017).

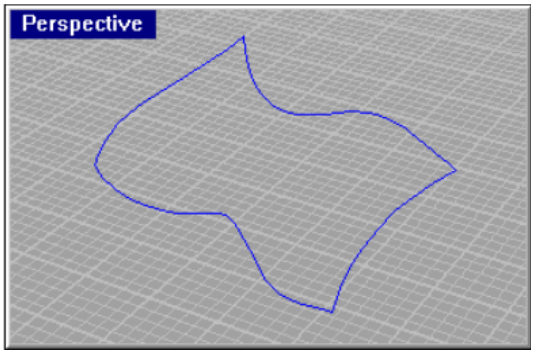


Fig 14 Superficie a partir de Aristas

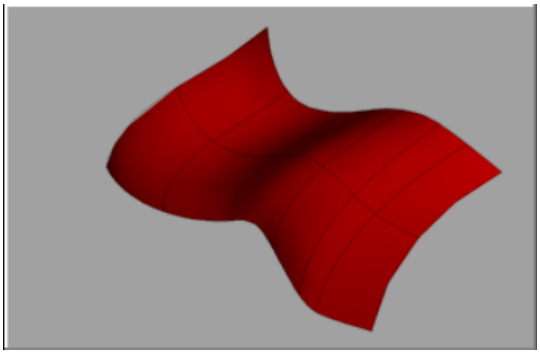


Fig 15 Superficie a partir de Aristas

Revolucionar una Curva Alrededor de un Eje: Cuando se revoluciona una curva se crea una superficie al dar vueltas una figura alrededor de un eje. El objeto resultante puede ser una superficie abierta o un sólido cerrado, dependiendo de si la curva está abierta o cerrada y de lo lejos que la hizo girar («Create surfaces | Rhino 3-D modeling» 2017).

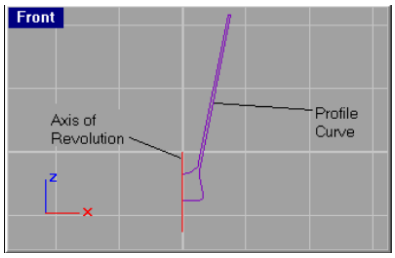


Fig 16 Revolucionar una curva alrdedeor de un Eje

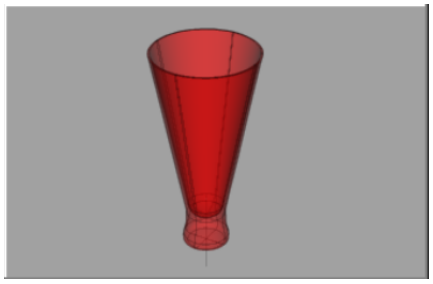


Fig 17 Revolucionar una curva alrdedeor de un Eje

Extruir una Curva en Línea Recta: Al extruir se crea una superficie a partir de una curva. Si la curva es plana, la extrusión será perpendicular, al plano de la curva. Si la curva no es plana, la dirección de la extrusión depende del plano de construcción activo («Create surfaces | Rhino 3-D modeling» 2017).

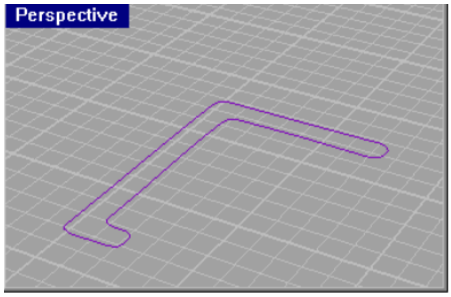


Fig 16 Extruir una curva en linia recta.

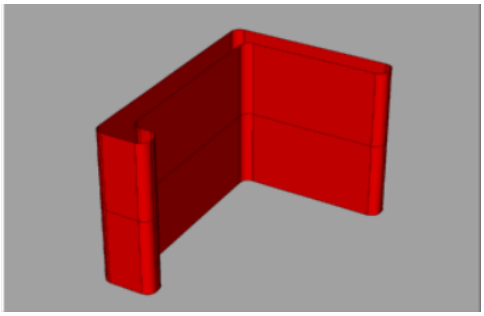


Fig 17 Extruir una curva en linia recta.

Barrido de una Curva por un solo Carril: Barrer crea una superficie con cortes que mantienen la orientación inicial de la forma de la(s) curva(s) a la curva de camino. Observe la diferencia entre la superficie creada con este método y el ejemplo anterior de la curva extruida que usó la misma curva de camino («Create surfaces | Rhino 3-D modeling» 2017).

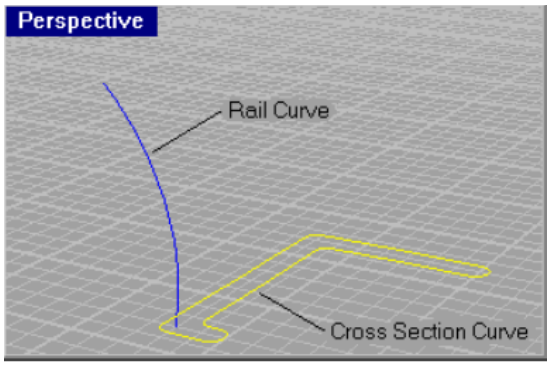


Fig 20 Barrido de una Curva por un sólo Carril

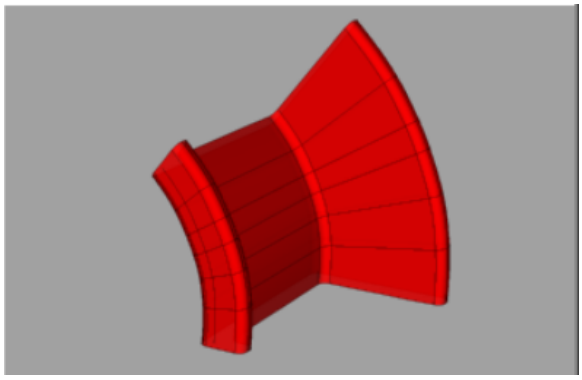


Fig 21 Barrido de una Curva por un sólo Carril

Barrido de un Curva por dos Carriles: Utilizar dos carriles para un barrido crea una superficie suave a través de dos o más formas de curvas que siguen dos carriles. Los carriles también afectan a la figura total de la superficie. Utilice este comando cuando quiera controlar la situación de los bordes de la superficie («Create surfaces | Rhino 3-D modeling» 2017).

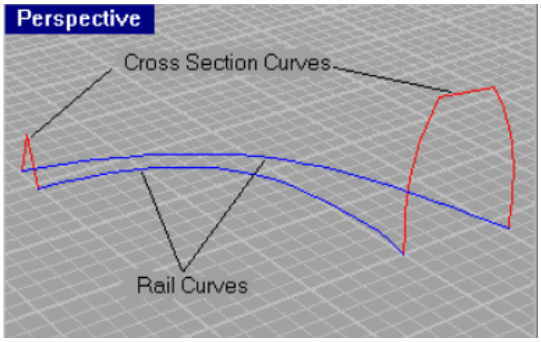


Fig 22 Barrido de un Curva por dos Carriles

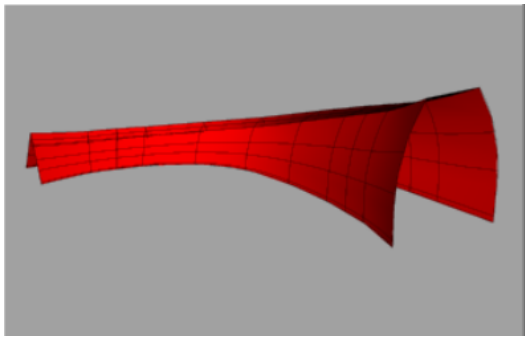


Fig 23 Barrido de un Curva por dos Carriles

Revolucionar una Curva por un Carril: Revolucionar un carril crea una superficie al hacer girar el perfil de una curva alrededor de un eje mientras al mismo tiempo se sigue la curva de carril. Esto es básicamente lo mismo que barrer a lo largo de dos carriles (Sweep Along 2 Rails), excepto que uno de los carriles sea el punto del centro.

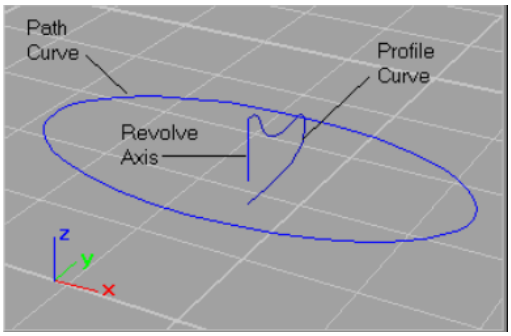


Fig 24 Revolucionar una Curva por un Carril

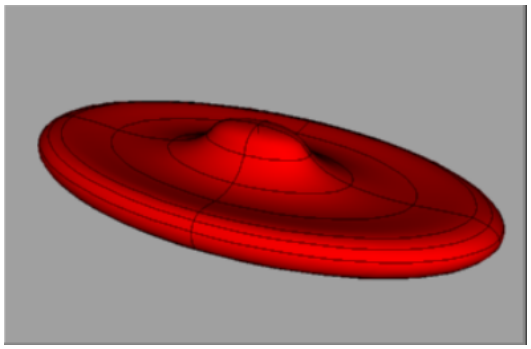


Fig 25 Revolucionar una Curva por un Carril

Elevar una Superficie a través de Curvas: La elevación (Loft) crea una superficie suave que se mezcla entre las formas curvas seleccionadas. Esta superficie es similar al ejemplo en la anterior sección, el Barrido de una Curva con Dos Carriles (Sweep a Curve with Two Rails), pero se crea sin curvas de carril. A cambio, los bordes de la superficie se crean ajustando curvas suaves a través de formas curvas («Create surfaces | Rhino 3-D modeling» 2017).

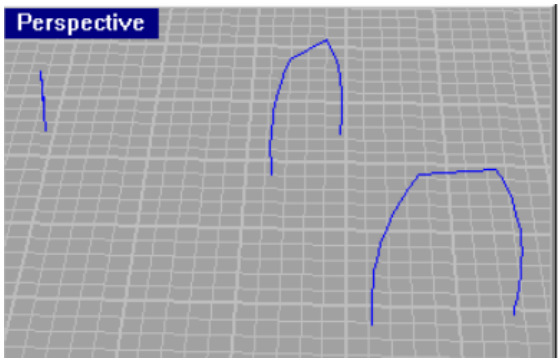


Fig 26 Elevar una Superficie a través de Curvas

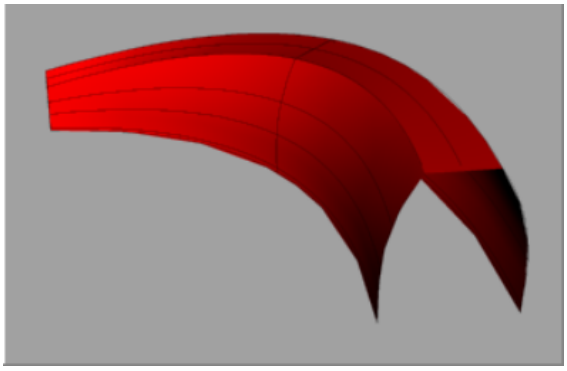


Fig 27 Elevar una Superficie a través de Curvas

Mezclar una Superficie entre Dos Superficies: Puede crear una mezcla entre dos superficies que se encontrará suavemente en los bordes de la superficie.

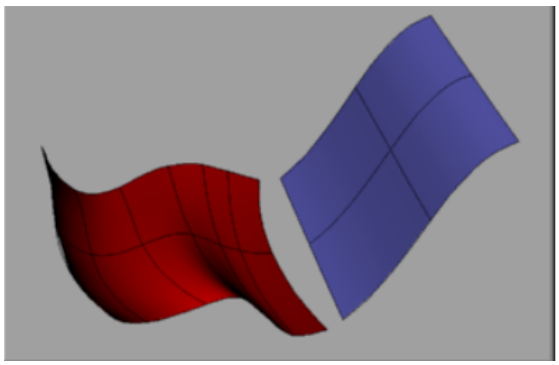


Fig 28 Mezclar una Superficie entre Dos Superficies:

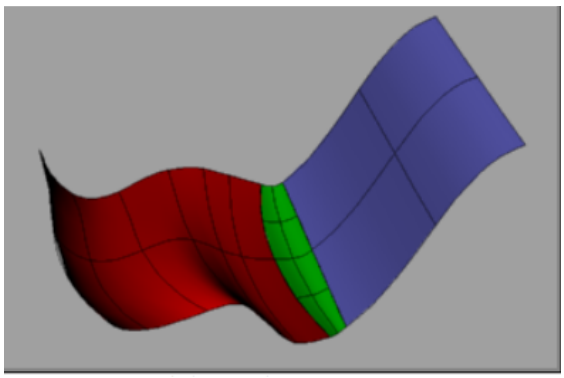


Fig 29 Mezclar una Superficie entre Dos Superficies:

1.4.1.4 AutoCAD

El modelado de superficies proporciona la capacidad de crear una forma más libre que el modelado sólido no puede proporcionar. AutoCad utiliza una manera de crear modelos de superficie con funcionalidades de extruir, girar, barrer, o perfiles de loft.

Creación de una superficie de **LOFT** como modelo de superficie asociativa (Shih 2014).

A Los perfiles transversales originales.

B El modelo después de usar el comando LOFT.

C El modelo después de editar uno de los perfiles transversales.

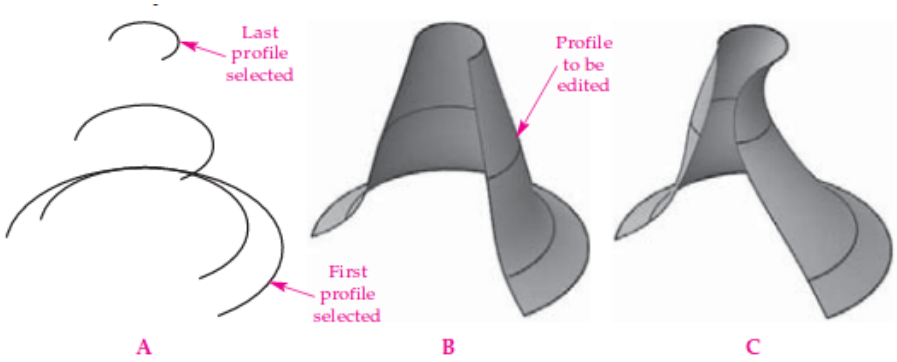


Fig 30 LOFT

Modificación de una superficie de **LOFT** creada a partir de perfiles cerrados (círculos) (Shih 2014).

A La superficie del desván original.

B El modelo después de editar el perfil cerrado en la parte superior.

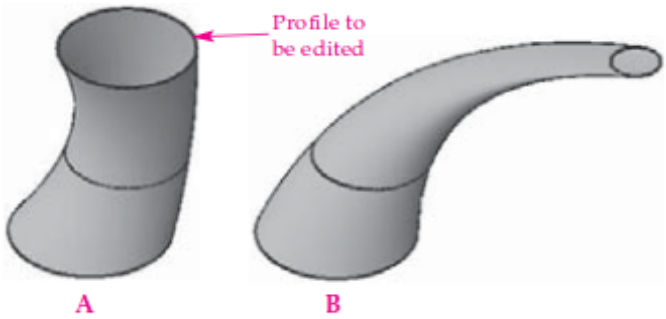


Fig 31 LOFT

Creación de una superficie de red.

Las splines se utilizan como perfiles en este ejemplo (Shih 2014).

A Los perfiles utilizados para definir la superficie se seleccionan en el orden numerado mostrado. Los perfiles 1-3 definen la primera dirección y se muestran en color. Los perfiles 4-8 definen la segunda dirección.

B El modelo superficial resultante

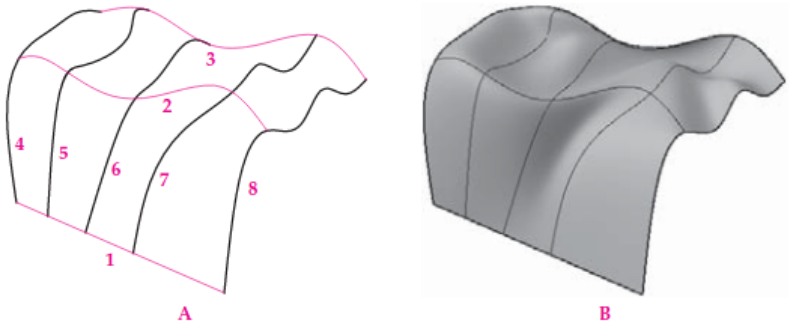


Fig 32 Creación de una superficie de red

Superficies de red (Shih 2014):

A Los perfiles 1-3 definen la primera dirección de la superficie y se muestran en color. Los perfiles 4 y 5 definen la segunda dirección de la superficie. El modelo de superficie resultante se muestra en forma de alambre y en forma sombreada.

B Los perfiles 1-4 definen la primera dirección de la superficie y se muestran en color. Los perfiles 5 y 6 definen la segunda dirección de la superficie. El modelo de superficie resultante se muestra en forma de alambre y en forma sombreada.

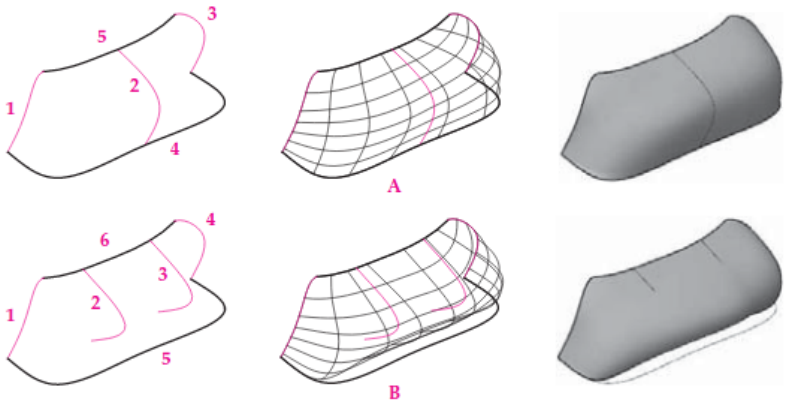


Fig 33 Superficies de red

Creación de una **superficie de red desde los bordes** de los objetos existentes (Shih 2014).

A Dos bordes de la región (1 y 2) se seleccionan para definir la primera dirección de la superficie. Un borde subobjeto de superficie (3) y un borde subobjeto sólido (4) se seleccionan para definir la segunda dirección de la superficie.

B El modelo de superficie resultante.

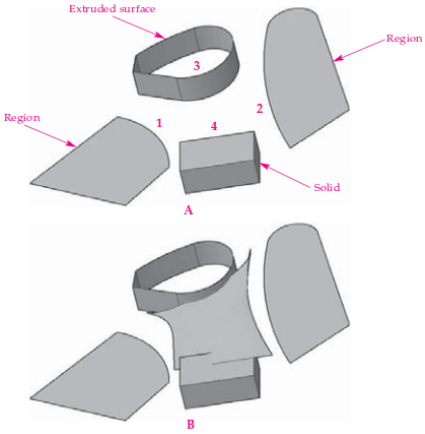


Fig 34 Superficie de red desde los bordes

Se crea una **superficie de mezcla** entre dos superficies existentes seleccionando un borde inicial y final (Shih 2014).

A Dos superficies de loft existentes.

B El modelo después de crear una superficie de mezcla con los valores predeterminados del comando SURFBLEND

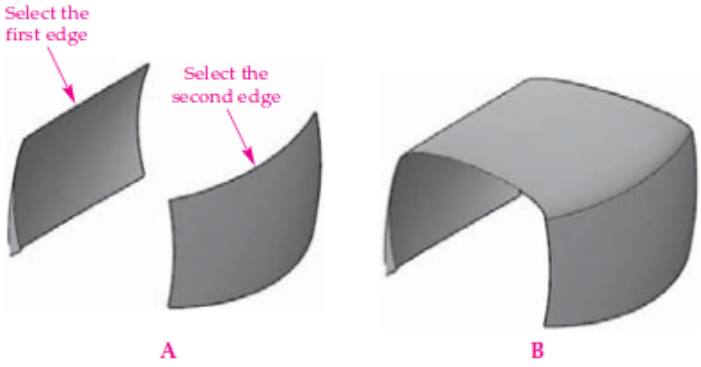


Fig 35 Superficie de mezcla

Edición de una **superficie NURBS** convertida desde una superficie de procedimiento (Shih 2014).

A El modelo original es una superficie de loft creada como una superficie de procedimiento.

B El modelo después de usar los comandos CONVTONURBS y CVSHOW.

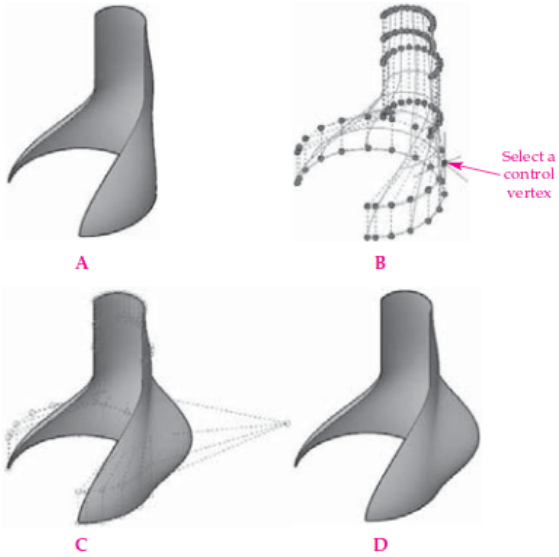


Fig 36 Superficie NURBS

1.4.2 Sistemas libres

1.4.2.1 Salome-Meca.

La funcionalidad filling de Salome-Meca hace un barrido entre dos o más curvas («SALOME Geometry User's Guide: Introduction to Geometry» [sin fecha]).

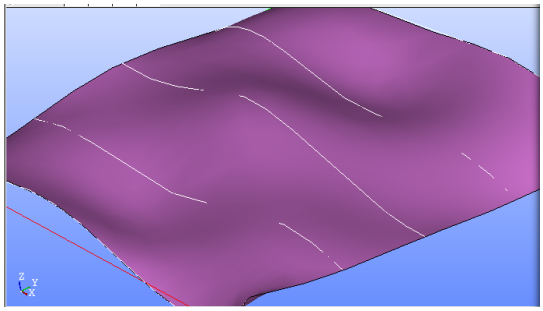


Fig 37 Filling

La funcionalidad smooth surface crea una superficie que pasa por los puntos deseados («SALOME Geometry User's Guide: Introduction to Geometry» [sin fecha]).

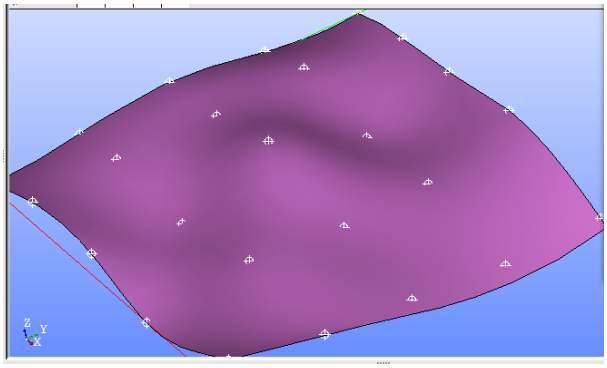


Fig 38 Smooth Surface

1.4.3 Consideraciones del análisis de las funcionalidades existentes

Las herramientas analizadas poseen funcionalidades que permiten modelar superficies, muchas de estas comunes entre ellas. La variante que más se destaca es el barrido de curvas siguiendo uno o más perfiles, permitiendo modelar superficies deseadas por el usuario. También, se evidencian otras funcionalidades que permiten la conformación de superficies como es el caso de la extrusión y la revolución. La aplicación Rhinoceros modela las superficies empleando interpolación de NURBS, pues pueden representar formas simples, como planos y cilindros, y también superficies de forma libre; lo que permite modificar las superficies empleando sus puntos de control, siendo esta vía la más interactiva a la hora de trabajar con estos objetos. La aplicación salome posee la funcionalidad de barrido de curvas que tienen las propietarias pero no se puede modificar sus valores después de creada la figura.

La selección de las funcionalidades básicas para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas, destinado a su integración en una aplicación CAD, tuvo su base en el análisis integral y reutilización de secciones de código fuente de las aplicaciones Freecad y del módulo “Geometry” de la aplicación “Salome-Meca”, en correspondencia con decisiones de proyecto en el grupo de investigación.

1.5 Metodologías para el 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 (RODRÍGUEZ SÁNCHEZ 2015).

1.5.1 AUP-UCI

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 decidio 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 (RODRÍGUEZ SÁNCHEZ 2015).

Descripción de las Fases:

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes tres fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre. Para una mayor comprensión se muestra la Tabla 5 del Anexo.

Descripción de las disciplinas:

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 7 disciplinas también, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en tres disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes tres disciplinas de AUP asociadas a la parte de gestión para la variación 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 Tabla 6 del Anexo (RODRÍGUEZ SÁNCHEZ 2015).

Descripción de los 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 Tabla 7 del Anexo.

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 (RODRÍGUEZ SÁNCHEZ 2015):

- ✔ Escenario 1: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta.
- ✔ Escenario 2: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que

ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio.

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

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

1.6 Herramientas y Tecnologías para el desarrollo

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

1.6.1 Herramienta de modelado

Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de funcionalidades que apoyan al desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (S. PRESSMAN 2003).

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Existe una alternativa libre y gratuita de este software, la versión Visual Paradigm UML 6.4 Community Edition (Community Edition, ya que existe la Enterprise, Professional, etc.). Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos.

Se caracteriza por (S. PRESSMAN 2003):

- ✔ Disponibilidad en múltiples plataformas (Windows, Linux).
- ✔ 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.
- ✔ Generación de código para Java y exportación como HTML.
- ✔ Soporte de UML versión 2.1.
- ✔ Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- ✔ Ingeniería inversa - Código a modelo, código a diagrama.
- ✔ Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.
- ✔ Generación de código - Modelo a código, diagrama a código.
- ✔ Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- ✔ Generación de código y despliegue de EJB - Generación de beans para el desarrollo y despliegue de aplicaciones.
- ✔ Diagramas de flujo de datos.
- ✔ Soporte ORM - Generación de objetos Java desde bases de datos.
- ✔ Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- ✔ Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- ✔ Generador de informes.
- ✔ Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.

1.6.2 Open CASCADE Technology

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

Alternativamente, Open CASCADE puede ser utilizada bajo los términos de licencia comercial o acuerdo contractual. Está diseñada para ser altamente portátil y es conocida por trabajar en una amplia gama de plataformas (UNIX, Linux, Windows, Mac OS X, Android). La versión (7.0.0. beta) está certificada oficialmente en las plataformas Windows (IA-32 y x86-64), Linux (x86-64), MAC OS X (x86-64) y Android (4.0.4 ARMv7) («Open CASCADE Technology: Overview» [sin fecha]). Open Cascade Community Edition (OCE, en español Edición de la Comunidad de Open Cascade) es una versión de OCCT en la que la comunidad del software libre aporta sus experiencias y recomendaciones de optimización a las versiones liberadas mediante foros o la página oficial de desarrollo de esta tecnología.

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

1.6.3 Lenguaje de programación

C++ es un lenguaje imperativo orientado a objetos derivado del C. En realidad, un superconjunto de C, que nació para añadirle cualidades y características de las que carecía. El resultado es que, como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción (STROUSTRUP. 2013). Estrictamente hablando, C no es un subconjunto de C++; de hecho, es posible escribir código C que es ilegal en C++. Pero a efectos prácticos, dado el esfuerzo de compatibilidad desplegado en su diseño, puede considerarse que C++ es una extensión del C clásico. La definición "oficial" del lenguaje nos dice que C++ es un lenguaje de propósito general basado en el C, al que se han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones inline, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de librería (STROUSTRUP. 2013).

Tanto C como C++ son lenguajes de programación de propósito general. Todo puede programarse con ellos, desde sistemas operativos y compiladores hasta aplicaciones de bases de datos y procesadores de texto, pasando por juegos, aplicaciones a medida, etc (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. También es el que se usa actualmente en el proyecto al que está integrado la investigación.

1.6.4 Framework de desarrollo

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 («About Qt - Qt Wiki» [sin fecha]). Está disponible bajo varias licencias: licencia comercial y para software libre con varias versiones de la GPL y la LGPL.Este es mucho más que un conjunto de herramientas de Interfaces Gráficas de Usuario (GUI, por sus siglas en inglés). Proporciona módulos para el desarrollo multiplataforma en las áreas de redes, bases de datos, OpenGL 11 , tecnologías web, sensores, protocolos de comunicación (Bluetooth, puertos serie), XML 12 y procesamiento JSON 13 , impresión, la generación de PDF, y mucho más («About Qt - Qt Wiki» [sin fecha]). El Entorno Integrado de Desarrollo recomendado para el empleo de este framework es el Qt Creator.

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

1.6.5 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 (Torvalds y Hamano 2010).

La característica Git que realmente hace que se aparte de casi todos los otros SCM es su modelo de ramificación. 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 (Torvalds y Hamano 2010):

- ✔ **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 fíjelo.
- ✔ **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 rama 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 este 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 presente investigación.

1.7 Conclusiones del capítulo

El capítulo desarrollado contiene los aspectos esenciales relacionados con la fundamentación teórica, como el perfil de la investigación y los estudios realizados sobre las funcionalidades para el modelado de superficies interpoladas de los sistemas para el Diseño Asistido por Computadoras, la metodología y las tecnologías de desarrollo.

Como resultado de este proceso se pudo arribar a las siguientes conclusiones:

1. Los sistemas propietarios AutoCAD, Solid Edge y CATIA modelan superficies interpoladas a partir de perfiles y caminos de construcción, sin embargo, Rhinoceros emplea NURBS, lo que permite la modificación a partir de sus puntos de control.
2. FreeCAD y LibreCAD no poseen las funcionalidades que tienen las herramientas propietarias para modelar superficies de curvas interpoladas. Salome-Meca posee las funcionalidades de suavizado de plano y barrido; para modificar las superficies en esta herramienta es necesario remitirse a los puntos que se encuentran en el árbol y actualizar los datos manualmente.

2 Propuesta de solución

En este capítulo se plantea la propuesta de solución que incluye el modelo del dominio, la descripción de las funcionalidades, los requisitos funcionales y no funcionales, las historias de usuario y aspectos de diseño (el estilo y patrón arquitectónico, los patrones de diseño, el modelo de clases y diagramas de secuencias).

2.1 Modelo del dominio

El modelo de dominio facilita la comprensión de los sistemas debido a que permite clasificar y agrupar objetos, disminuyendo la complejidad y el número de elementos en el modelado. También, se utiliza para describir los objetos y sus relaciones con los sistemas, facilitando una mejor comunicación entre usuarios, desarrolladores y clientes (ERIKSSON y PENKER. 2000).

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

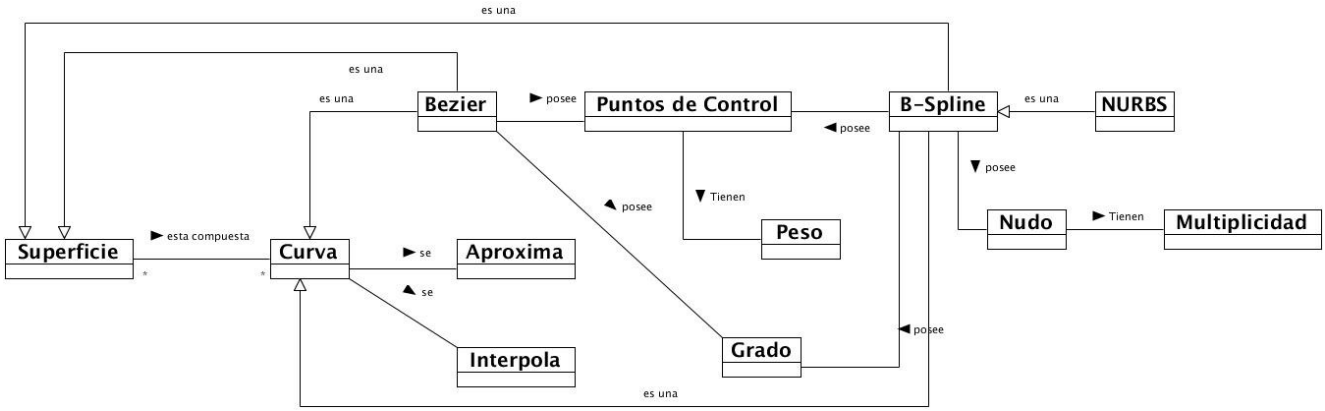


Fig 39 Diagrama del modelo de dominio

Una superficie está compuesta por curvas. Estas pueden ser de tipo Bézier, B-Spline o NURBS, esta última es un tipo de B-Spline. Las curvas de Bézier están compuestas por puntos de control que es por donde pasa dicha curva y los pesos asociados a ellos definen cuánto afectan a la curva. Las NURBS y B-Spline están compuestas por puntos de control, pesos asociados a estos, nudos con sus multiplicidades y el grado de la curva. Las curvas se pueden crear por interpolación o aproximación, donde por interpolación la curva pasa por cada punto de control y por aproximación pasa cerca de estos pero no los tocan. Las superficies cumplen con los requisitos ya mencionados.

2.2 Descripción de las funcionalidades

Las funcionalidades serán diseñadas para el modelado de piezas tipo chapa con perfiles curvos, superficies NURBS y B-Splines. Estas estarán integradas a la aplicación CAD que está desarrollando el

grupo de investigación SIPII la cual poseerá una sesión “Curved Plate” donde se encontraran estas funcionalidades. Al activarse este modo se mostrarán una serie de botones que tendrán asociados los algoritmos necesarios para realizar las transformaciones pertinentes a este tipo chapas. Un primer botón llamado “Filling” permitirá crear una cara curvilínea desde varios bordes. “Smoothing” permitirá el suavizado de superficie dada una lista de puntos. “Bezier” permitirá crear una superficie de Bézier dado sus parámetros de construcción. “B-Spline” permitirá crear una superficie B-Spline dado sus parámetros de construcción. “NURBS” permitirá crear una superficie B-Spline racional no uniforme dado sus parámetros de construcción.

2.3 Requisitos

“Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información” (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” (SOMMERVILLE. 2006).

A continuación, se enumeran los requisitos funcionales definidos para el módulo:

RF 1. Modelar relleno de superficie.

RF 2. Modelar un suavizado de superficie.

RF 3 Modelar una superficie de Bezier.

RF 4 Modelar una superficie B-Spline.

RF 5 Modelar una superficie NURBS.

2.3.2 Requisitos No Funcionales

Los requisitos no funcionales son “restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema” (SOMMERVILLE. 2006).

A continuación, se exponen los requisitos no funcionales del componente:

Portabilidad:

RNF 1. Software: SO.: GNU-Linux, GCC: 4.2.4

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).

Tabla 1: Historia de Usuario Suavisado de Superficie

Número: 1	**Nombre del requisito:** Suavizado de superficie.
Programador: Manuel Fornés Martínez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 2semanas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 1, 5 semanas

Descripción:

1- Objetivo:

Permitir el suavizado de superficie.

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

Para crear el suavizado de superficie se debe tener en cuenta los siguientes datos::

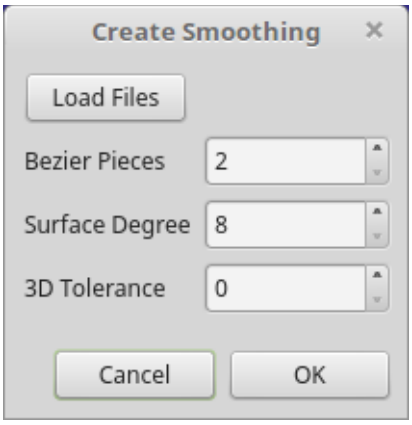
- La lista de puntos de los cuales se aproxima.
- Número máximo de piezas de Bézier en la superficie resultante.
- Max BSpline grado de superficie de la superficie resultante Bspline.
- Tolerancia 3D de la aproximación inicial.

3- Flujo de la acción a realizar:

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción Apply se crea el suavizado de la superficie.
- Si se selecciona el botón Apply and Close se crea el suavizado de la superficie y se cierra la ventana.
- Si se selecciona el botón Close se cierra la ventana.

Observaciones:

Prototipo de interfaz:



2.4 Diseño

"La esencia del diseño del software es la toma de decisiones sobre la organización lógica del software. Algunas veces, se representa esta organización lógica como un modelo en un lenguaje definido de modelado tal como UML y otras veces simplemente se utiliza notaciones informales y esbozos para representar el diseño" (SOMMERVILLE. 2006). El proceso de diseño tiene asociado la decisión del tipo arquitectura y los patrones de diseño que empleará el sistema, así como la confección de distintos diagramas que favorezcan el trabajo en la fase de implementación.

2.4.1 Estilo y patrón arquitectónico del software

"Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una estructura para todos los componentes del sistema. En caso de que una arquitectura existente se vaya a someter a reingeniería, la imposición de un estilo arquitectónico desembocará en cambios fundamentales en la estructura del software, incluida una reasignación de la funcionalidad de los componentes" (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" (PRESSMAN. 2005).

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.

2.4.1.1 Arquitectura en Capas

El modelo en capas organiza el sistema en capas, cada una de las cuales proporciona un conjunto de servicios a la capa superior. Este modelo soporta el desarrollo incremental del sistema, los cambios y es portable. Además, cuando las interfaces de las capas cambian o se añaden nuevas funcionalidades

a una capa, solamente se ven afectadas las capas adyacentes. Entre las desventajas de este modelo es que la estructuración de los sistemas puede resultar difícil y el rendimiento se puede ver afectado, pues se puede incurrir en que una funcionalidad debe pasar por muchas capas para alcanzar la capa superior que solicitó su servicio («OpenGL Overview» [sin fecha]). Se decide el empleo de este modelo en la confección del módulo porque soporta bien los cambios y el desarrollo incremental.

El diseño arquitectónico más apropiado para el desarrollo de las funcionalidades para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas a ser empleadas en sistemas de Diseño Asistido por Computadora, debe basarse en lo fundamental, en una Arquitectura por capas que permita la comunicación con el núcleo de la aplicación y con otros módulos de la misma.

2.4.2 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 interfaces.

Los patrones GRASP definidos en las funcionalidades:

- ✔ **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 BezierSurface, BSplineSurface, NurbSurface, SmoothingSurface y FillingSurface, que se encargan de la creación de las superficies.
- ✔ **Creador**: permite crear objetos de una clase determinada (Larman 2003). Este patrón se evidencia en las clases de interfaz DlgBezierSurface, DlgBSplineSurface, DlgNurbSurface, DlgSmoothingSurface y DlgFillingSurface ya que tiene la información necesaria para la creación de objetos tipo BezierSurface, BSplineSurface, NurbSurface, SmoothingSurface y FillingSurface.
- ✔ **Polimorfismo**: se emplea para asignar comportamientos distintos según el tipo de clase (Larman 2003). Se evidencia en las clases BezierSurface, BSplineSurface, NurbSurface, SmoothingSurface y FillingSurface, que heredarán de la clase Feature y reimplementan los métodos execute, onChanged, mustExecute y getViewProviderName.
- ✔ **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 (Larman 2003).

Patrón GOF definido en las funcionalidades:

- ✔ ***Singleton***: Mediante su uso se garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella (Larman 2003). Este patrón se evidencia en las clases de

interfaz DlgBezierSurface, DlgBSplineSurface, DlgNurbSurface, DlgSmoothingSurface y DlgFillingSurface.

2.4.3 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).

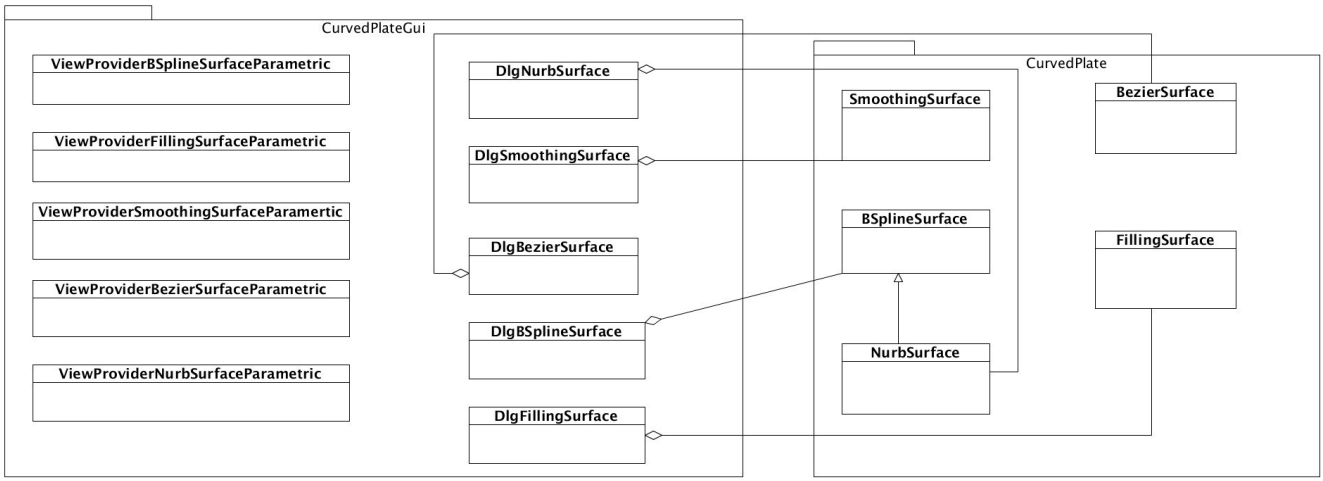


Fig 40 Diagrama de clase del diseño

2.4.4 Diagramas de secuencia del diseño

Un diagrama de secuencia muestra un conjunto de mensajes, dispuestos en una secuencia temporal; puede mostrar un escenario, es decir, una historia individual de una transacción. Uno de sus usos es mostrar la secuencia del comportamiento de un caso de uso. Cuando está implementado el comportamiento, cada mensaje en un diagrama de secuencia corresponde a una operación en una clase, a un evento disparador, o a una transición en una máquina de estados (JACOBSON, RUMBAUGH y BOOCH. 2000).

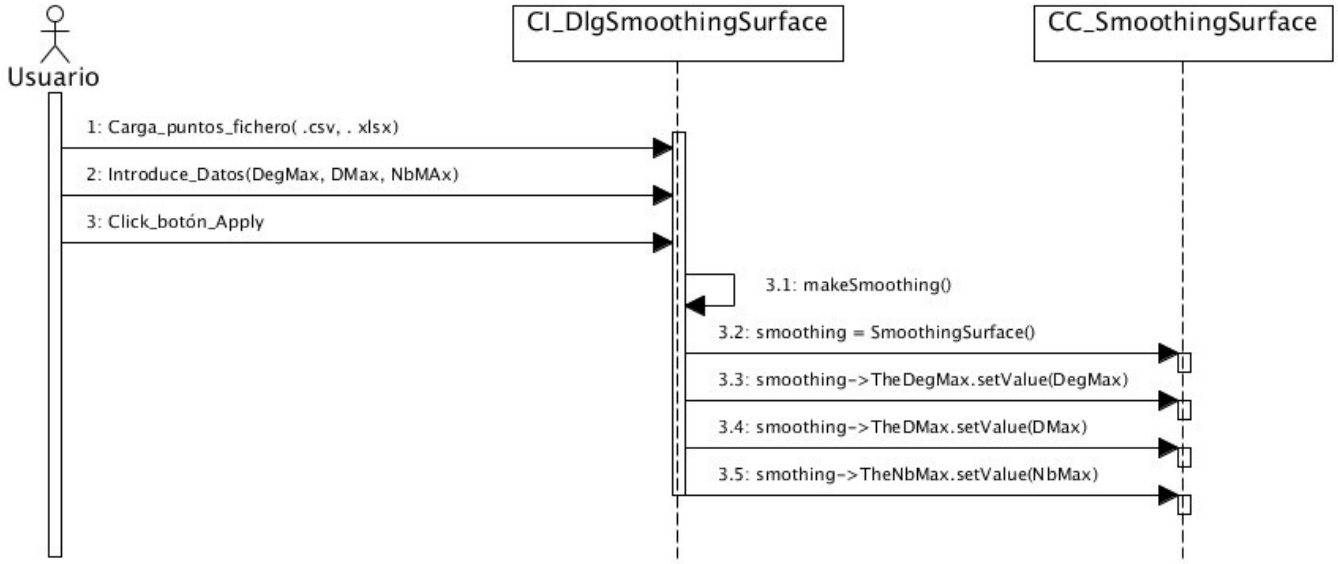


Fig 41 Diagramas de secuencia del diseño SmoothingSurface

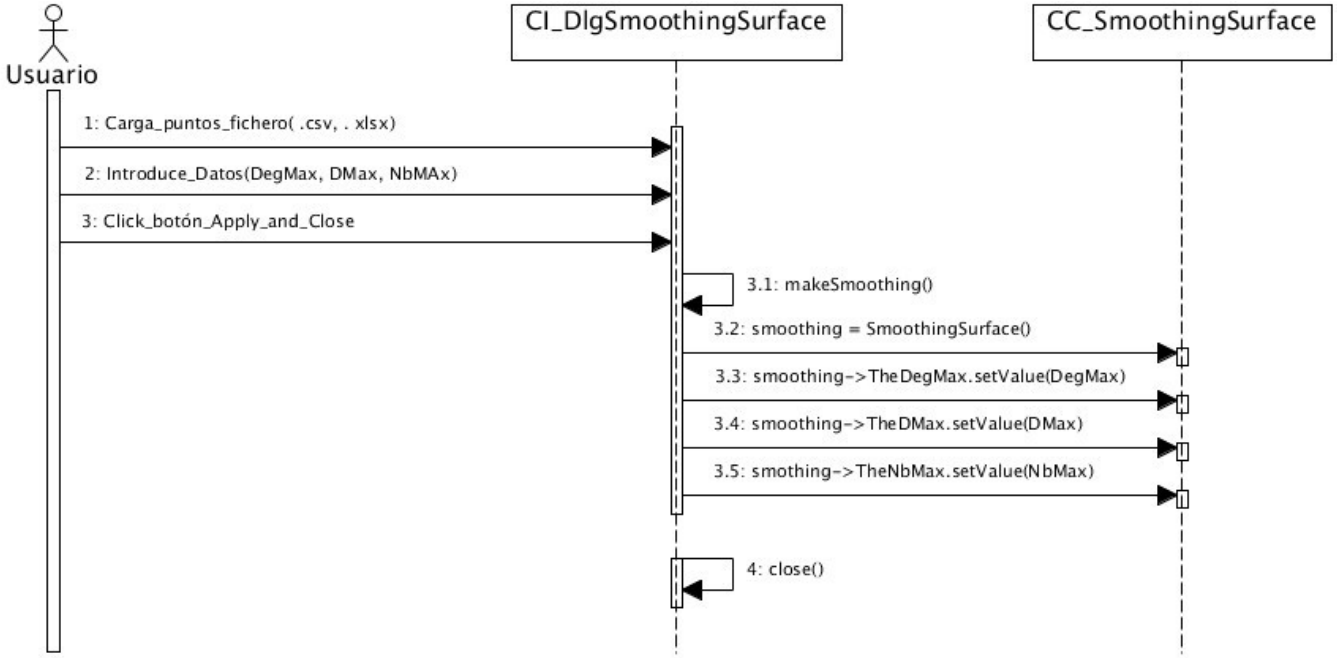


Fig 42 Diagramas de secuencia del diseño SmoothingSurface

2.5 Conclusiones del capítulo

En el presente capítulo quedó descrito la propuesta de solución, para la cual se realizaron los artefactos definidos en la fase de diseño de la metodología AUP-UCI, los cuales permiten una caracterización de la propuesta con el fin de garantizar una correcta documentación para la posterior referencia y continuidad de desarrollo, así como sentar las bases metodológicas para la siguiente fase de desarrollo (implementación). Además, se identificó como patrón arquitectónico a emplear en capa, pues soporta bien los cambios y el desarrollo incremental.

3 Implementación y Pruebas

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 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

Tabla 2: Estándar de codificación

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.	vrtual ~Foo()
La declaración de funciones o métodos siempre comenzarán con letra inicial minúsculas. En caso de ser un nombre compuesto se regirá por la normativa	<Tipo dato retorno> funcion() <Tipo dato retorno> funcionCompuesta()

CamelCase-lowerCamelCase.	<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> artibutoNombreCompuesto;
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 minúsculas separados a un espacio cada uno y en caso de ser compuesto utilizar normativa CamelCase-lowerCamelCase.	<Tipo dato retorno> funcion(<tipo><id1>, <tipo><id2>, <tipo><id3>)
Los identificadores de los parámetros dentro de los constructores de las clases deben estar escritos con minúsculas separados a un espcio 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 estandar determinado por el framework de Qt.	Para las estructuras if, else, if else:
	
	<estructura de control>(condición){
	
	Tarea a ejecutar
	
	}
	
	Para los bucles while, for, do while y otros:
	
	<bucle>(condiciones){
	
	tarea a ejecutar
	
	}

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

3.1.2 Diagrama de componente

Un diagrama de componentes muestra dependencias entre los componentes, que no son más que una unidad física de implementación con interfaces bien definidas pensada para ser utilizada como parte reemplazable de un sistema. Puede mostrar un sistema configurado, con la selección de componentes

usados para construirlo o un conjunto de componentes disponibles (una biblioteca de componentes) con sus dependencias (JACOBSON, RUMBAUGH y BOOCH. 2000).

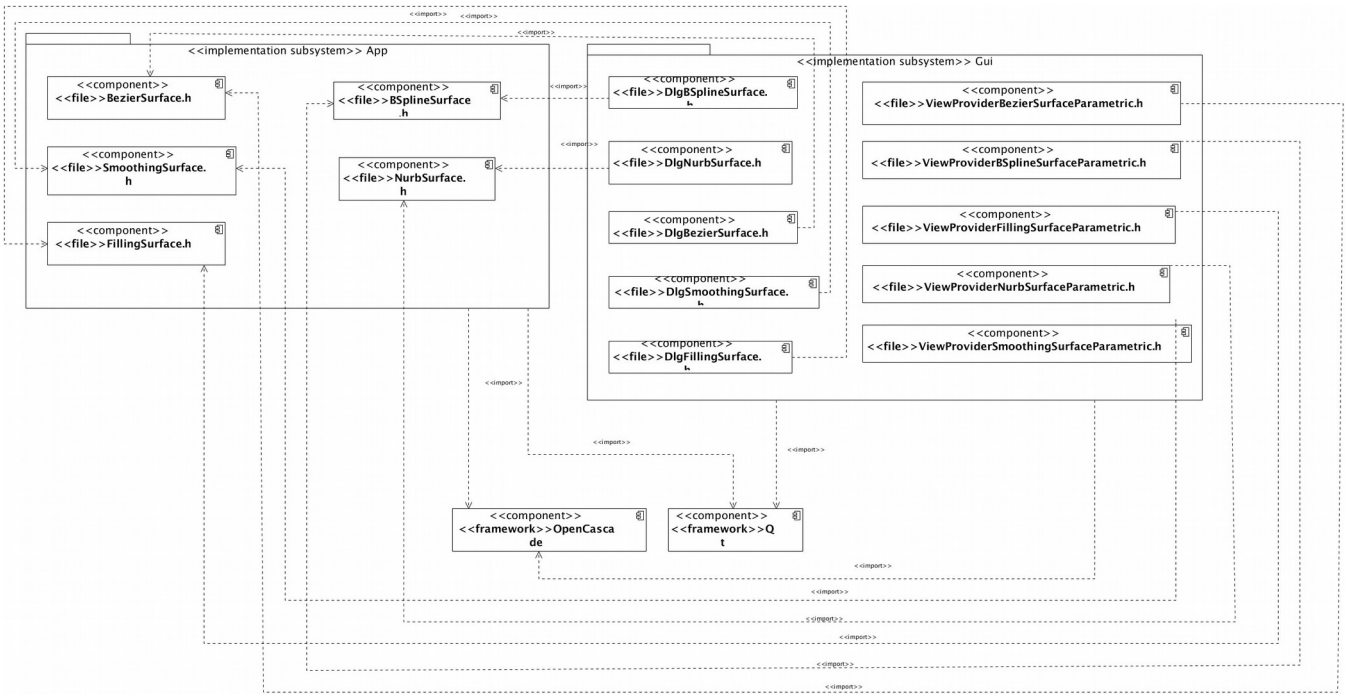


Figura 41 Diagrama de componentes

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 prueba

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

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

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

3.2.3 Diseño de Casos de Prueba

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

Tabla 3: Diseño de Casos de Prueba

Descripción general			
Permitir la creación de un suavizado de superficie dado una lista de puntos.			
SC 1 Modelar un suavizado de superficie			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción de Load File.	Selecciona la opción de Load File el cual mostrará una ventana para elegir el fichero con los puntos, dicho fichero de extensión .csv o .xlsx.	Brinda la posibilidad de crear una superficie con los puntos cargados del fichero.	CurvedPlate/Smoothing/Load File
EC 1.2 Opción de Apply.	Selecciona la opción de Apply.	Crea la Superficie con los datos de entrada.	CurvedPlate/Smoothing/Apply
EC 1.3 Opción de Apply and Close.	Selecciona la opción de Apply and Close.	Crea la Superficie con los datos de entrada y cierra la ventana.	CurvedPlate/Smoothing/Apply and Close
EC 1.4 Opción de Close.	Selecciona la opción de Close.	Cierra la ventana.	CurvedPlate/Smoothing/Close

3.2.4 Pruebas Unitarias

Para la realización de las pruebas unitarias se empleó Qt Test, el cual es un framework para pruebas de este tipo a aplicaciones y librerías. “Qt Test provee todas las funcionalidades comunes de los framework de pruebas unitarias, así como extensiones para probar interfaces gráficas de usuario” («Qt Test Overview | Qt Test 5.9» [sin fecha]). A continuación, se muestra una imagen de las pruebas unitarias realizadas durante el proceso de implementación, en la que se evidencia la aceptación de todas las verificaciones:

```
********* Start testing of CurvedPlateTest *********
Config: Using QtTest library 5.5.1, Qt 5.5.1 (x86_64-lit
PASS   : CurvedPlateTest::initTestCase()
PASS   : CurvedPlateTest::makeBezierSurface()
PASS   : CurvedPlateTest::makeBSplineSurface()
PASS   : CurvedPlateTest::makeNurbSurface()
PASS   : CurvedPlateTest::makeSmootingSurface()
PASS   : CurvedPlateTest::makeFillingSurface()
PASS   : CurvedPlateTest::loadFileCSV()
PASS   : CurvedPlateTest::loadFileXLSX()
PASS   : CurvedPlateTest::cleanupTestCase()
Totals: 9 passed, 0 failed, 0 skipped, 0 blacklisted
********* Finished testing of CurvedPlateTest *********
```

Fig 42 Pruebas unitaria

3.2.5 Resultados de las pruebas

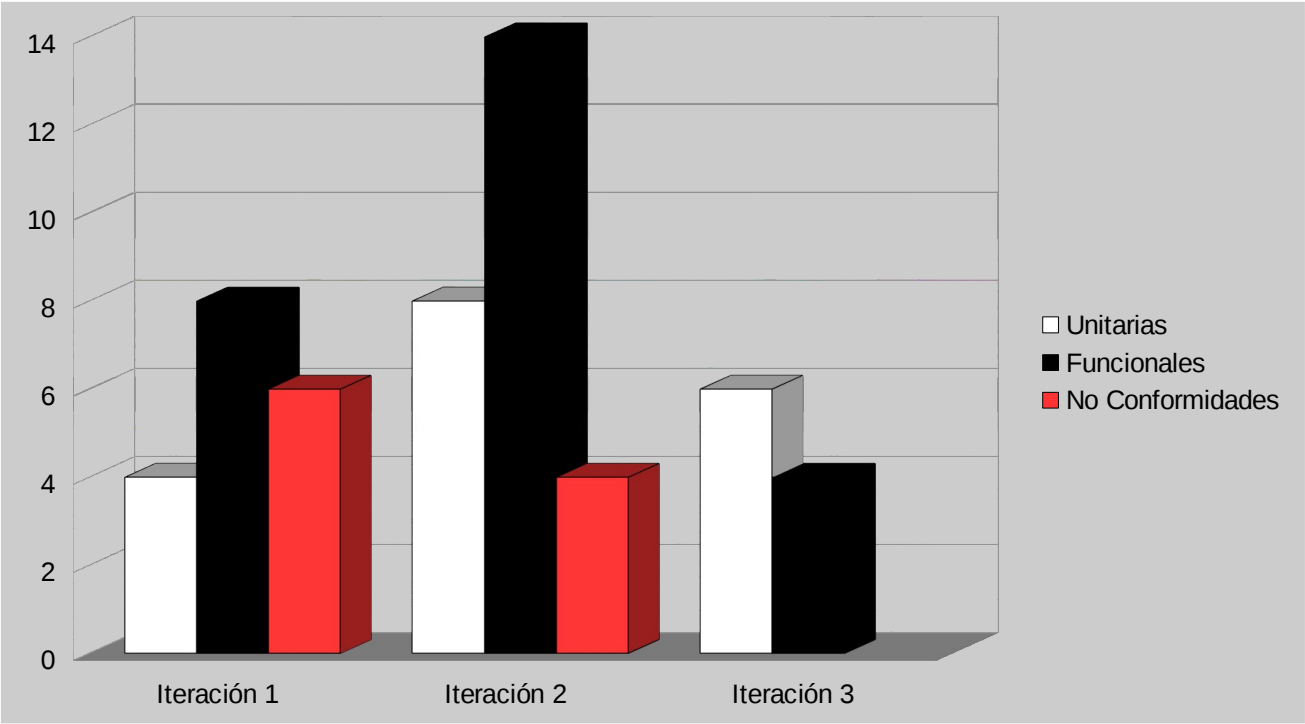
La realización de pruebas funcionales permitió identificar una serie de no conformidades:

Tabla 4: No Conformidades

No. NC	Requisito Funcional	Descripción	Complejidad	Estado
1	RF 1	La superficie no barre con la dirección de la curva.	Alta	Resuelta
2	RF 2	El plano no pasa por los puntos dados como restricciones de este.	Alta	Resuelta
3	RF 2	No se actualizan los puntos por donde pasa el plano.	Baja	Resuelta
4	RF 3	No se actualizan los puntos por donde pasa el plano.	Baja	Resuelta
5	RF 3	No se puede modificar los parámetros de construcción de la superficie.	Baja	Resuelta
6	RF 4	No se actualizan los puntos por donde pasa el plano.	Alta	Resuelta
7	RF 4	No se puede modificar los parámetros de construcción de la superficie.	Baja	Resuelta
8	RF 4	No se maneja los errores al construir la superficie.	Baja	Resuelta
9	RF 5	No se actualizan los puntos por donde pasa el plano.	Baja	Resuelta

10	RF 5	No se puede modificar los parámetros de construcción de la superficie.	Baja	Resuelta

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



La eficacia de las funcionalidades implementadas para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas, está avalada por los resultados de las pruebas realizadas, que demostraron el correcto funcionamiento de cada una de ellas.

3.3 Conclusiones del capítulo

En el presente capítulo se caracterizaron los estándares de codificación establecidos por el grupo de investigación para la implementación de las clases y los métodos definidos en la fase de diseño. En la etapa de implementación se realizaron pruebas unitarias a los métodos y se compararon los modelos obtenidos de algunas de las funcionalidades correspondientes a la propuesta de solución con los de SalomeMeca, con el objetivo de verificar la visualización obtenida. Además, se realizaron pruebas funcionales utilizando la técnica de pruebas de caja negra, las cuales permitieron detectar un total de 10 no conformidades que fueron corregidas, garantizando una mejor calidad de la solución.

4 Conclusiones generales

Como resultado del proceso de Investigación-Desarrollo realizado se arribó a las siguientes conclusiones generales:

- ✔ La selección de las funcionalidades básicas para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas, destinado a su integración en una aplicación CAD, tuvo su base en el análisis integral y reutilización de secciones de código fuente de las aplicaciones Freecad y del módulo “Geometry” de la aplicación “Salome-Meca”, en correspondencia con decisiones de proyecto en el grupo de investigación.
- ✔ El diseño arquitectónico más apropiado para el desarrollo de las funcionalidades para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas a ser empleadas en sistemas de Diseño Asistido por Computadora, debe basarse en lo fundamental, en una Arquitectura por capas que permita la comunicación con el núcleo de la aplicación y con otros módulos de la misma.
- ✔ La eficacia de las funcionalidades implementadas para el modelado de piezas tipo chapa con perfiles de curvas y superficies interpoladas, está avalada por los resultados de las pruebas realizadas, que demostraron el correcto funcionamiento de cada una de ellas.

5 Recomendaciones

A partir del estudio realizado y luego de haber analizado los resultados obtenidos se recomiendan los siguientes elementos a tener en cuenta para futuros trabajos:

- Incorporar las funcionalidades para modificar las piezas tipo chapa con perfiles de curvas y superficies interpoladas, mediante la aplicación de perforaciones, pestañas y otros detalles morfológicos de ese tipo de construcciones.
- Desarrollar las funcionalidades de ayuda al modelado (puntos, líneas y planos de construcción) que no fueron abordadas en el trabajo.
- Diseñar interfaces que propicien la explotación fácil y eficiente de las funcionalidades desarrolladas.

6 Referencias bibliográficas

1. About Qt - Qt Wiki. [en línea], [sin fecha]. [Consulta: 30 junio 2017]. Disponible en: https://wiki.qt.io/About_Qt.

2. AGOSTON, M.K., 2005. *Computer Graphics and Geometric Modelling* [en línea]. S.l.: Springer. Computer Graphics and Geometric Modeling. ISBN 978-1-85233-818-3. Disponible en: https://books.google.com.cu/books?id=fGX8yC-4vXUC.

3. CENTER, T.J.W.I.R., RAROUKI, R.T. y RAJAN, V.T., 1987. *Algorithms for Polynomials in Bernstein Form* [en línea]. S.l.: IBM Thomas J. Watson Research Division. Disponible en: https://books.google.com.cu/books?id=L5fiHgAACAAJ.

4. CHUI, C.K., 1988. *Multivariate Splines* [en línea]. S.l.: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104). CBMS-NSF Regional Conference Series in Applied Mathematics. ISBN 978-1-61197-017-3. Disponible en: https://books.google.com.cu/books?id=mIt4tPgq37oC.

5. CIDONCHA, M.G. del R., LOMAS, M.E.M., PALACIOS, J.M. y DÍAZ, S.P., 2007. *El libro de Catia V5: Módulos Part Design, Wireframe and Surface Design, Assembly Design y Drafting*. S.l.: Editorial Tebar. ISBN 978-84-7360-249-5.

6. Computacion Grafica: CURVAS NURBS. [en línea], [sin fecha]. [Consulta: 28 junio 2017]. Disponible en: http://vale085.blogspot.com/2007/04/curbas-nurbs.html.

7. CORMEN, T.H., LEISERSON, C.E., RIVEST, R.L. y STEIN, C., 2001. *Introduction To Algorithms* [en línea]. S.l.: MIT Press. ISBN 978-0-262-03293-3. Disponible en: https://books.google.com.cu/books?id=NLngYyWFl_YC.

8. Create surfaces | Rhino 3-D modeling. [en línea], 2017. [Consulta: 22 junio 2017]. Disponible en: http://docs.mcneel.com/rhino/5/help/en-us/seealso/sak_surface.htm.

9. DO CARMO, M.P., 1990. *Geometría diferencial de curvas y superficies*. S.l.: Alianza Editorial.

10. ERIKSSON, H.-E. y PENKER., M., 2000. *Business Modeling with UML: Business Patterns at Work*. S.l.: s.n.

11. GOLDMAN, R.N. y LYCHE, T., 1993. *Knot Insertion and Deletion Algorithms for B-Spline Curves and Surfaces:* [en línea]. S.l.: Society for Industrial and Applied Mathematics. Geometric design publications. ISBN 978-1-61197-158-3. Disponible en: https://books.google.com.cu/books?id=xCLmAgAAQBAJ.

12. JACOBSON, I., RUMBAUGH, J. y BOOCH., G., 2000. *El Lenguaje Unificado de Modelado. Manual de Referencia.* S.l.: s.n.

13. JEFFRIES, R., ANDERSON, A. y HENDRICKSON., C., 2001. *Extreme Programming Installed*. S.l.: s.n.

14. LARMAN, C., 2003. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado*. S.l.: Pearson Educación. ISBN 978-84-205-3438-1.

15. Open CASCADE Technology. [en línea], [sin fecha]. [Consulta: 30 junio 2017]. Disponible en: https://www.opencascade.com/doc/occt-6.9.1/refman/html/index.html.

16. Open CASCADE Technology: Overview. [en línea], [sin fecha]. [Consulta: 30 junio 2017]. Disponible en: https://www.opencascade.com/doc/occt-7.0.0/overview/html/index.html.

17. OpenGL Overview. [en línea], [sin fecha]. [Consulta: 29 junio 2017]. Disponible en: https://www.opengl.org/about/.

18. O'ROURKE, J., 1998. *Computational Geometry in C* [en línea]. S.l.: Cambridge University Press. Cambridge Tracts in Theoretical Computer Science. ISBN 978-0-521-64976-6. Disponible en: https://books.google.com.cu/books?id=gsv7HALW2jYC.

19. PIEGL, L. y TILLER, W., 2012. *The NURBS Book* [en línea]. S.l.: Springer Berlin Heidelberg. Monographs in Visual Communication. ISBN 978-3-642-97385-7. Disponible en: https://books.google.com.cu/books?id=CkeqCAAAQBAJ.

20. PRAUTZSCH, H., BOEHM, W. y PALUSZNY, M., 2013. *Bézier and B-Spline Techniques* [en línea]. S.l.: Springer Berlin Heidelberg. Mathematics and Visualization. ISBN 978-3-662-04919-8. Disponible en: https://books.google.com.cu/books?id=fEiqCAAAQBAJ.

21. PREPARATA, F.P. y SHAMOS, M., 2012. *Computational Geometry: An Introduction* [en línea]. S.l.: Springer New York. Monographs in Computer Science. ISBN 978-1-4612-1098-6. Disponible en: https://books.google.com.cu/books?id=_p3eBwAAQBAJ.

22. PRESSMAN., R.S., 2005. *Ingeniería del software. Un enfoque práctico. 6ta Edición.* S.l.: s.n.

23. Qt Test Overview | Qt Test 5.9. [en línea], [sin fecha]. [Consulta: 29 junio 2017]. Disponible en: http://doc.qt.io/qt-5/qtest-overview.html.

24. RODRÍGUEZ SÁNCHEZ, T., 2015. *Metodología de desarrollo para la actividad productiva de la UCI.* S.l.: s.n.

25. ROGERS, D.F., 2001. *An Introduction to NURBS: With Historical Perspective* [en línea]. S.l.: Morgan Kaufmann Publishers. Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. ISBN 978-1-55860-669-2. Disponible en: https://books.google.com.cu/books?id=DiyxPUiKvB8C.

26. RUIZ TENORIO, R., 2010. *Las Pruebas de Software y su Importancia en las Organizaciones.* S.l.: s.n.

27. S. PRESSMAN, R., 2003. *Ingeniería de Software. Un enfoque práctico. 5ta Edición.* S.l.: s.n.

28. SALOME Geometry User's Guide: Introduction to Geometry. [en línea], [sin fecha]. [Consulta: 30 junio 2017]. Disponible en: http://docs.salome-platform.org/7/gui/GEOM/index.html.

29. SHIH, R., 2014. *AutoCAD 2015 Tutorial - Second Level: 3D Modeling*. S.l.: SDC Publications. ISBN 978-1-58503-865-7.

30. SOMMERVILLE., I., 2006. *Ingeniería de Software. 8va Edición*. S.l.: s.n.

31. STROUSTRUP., B., 2013. *The C++ Programming Lenguage*. S.l.: s.n.

32. STRUIK, D.J., 2012. *Lectures on Classical Differential Geometry: Second Edition* [en línea]. S.l.: Dover Publications. Dover Books on Mathematics. ISBN 978-0-486-13818-3. Disponible en: https://books.google.com.cu/books?id=JVC8AQAAQBAJ.

33. TORVALDS, L. y HAMANO, J., 2010. Git: Fast version control system. *http://git-scm.com*,

34. YAMAGUCHI, F., 2012. *Curves and Surfaces in Computer Aided Geometric Design* [en línea]. S.l.: Springer Berlin Heidelberg. ISBN 978-3-642-48952-5. Disponible en: https://books.google.com.cu/books?id=OAOrCAAAQBAJ.

35. TORVALDS, L. y HAMANO, J., 2010. Git: Fast version control system. http://git-scm.com