UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 4

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Componente de intersección de curvas paramétricas en 2D para AsiXMec 2.0

Autora: Lázara Dalia Monteagudo Campos

Tutor: Ing. Angel Ulise Tabares González

Tutor: Ing. Karel Piorno Charchabal

Co-Tutor: Ing. Victor Alejandro Roque Dominguez

La Habana, julio 2017

"La posibilidad de realizar un sueño es lo que hace la vida interesante"

Paulo Coelho.

Dedicatoria

A mis padres: Lidia y Walberto.
A mis sobrinos Sofía, Fernando y Dairán.

Agradecimientos

A mis padres que me han apoyado en mis estudios.

A mi novio Dariel.

A mis tutores Angel Ulise, Victor y Karel por guiarme durante todo este curso que estuvimos compartiendo.

A Dianne Cordero, que sin ser mí tutora, me dio su apoyo.

Al profesor Millet que desde 1er año ha sido una persona con la que se puede contar.

Al profesor Andy H. que me brindado su ayuda.

A mis amigas: Yeinelis, Daylen, Dianne Cordero, Anelis, Ofelia y Yaima que me hemos compartido buenos momentos.

A mis compañeras del apartamento 90 105.

A mis compañeros del grupo cuatro.

A Yurisbel Bernal y Henry González que ha sido un verdadero amigo.

También a personas muy queridas de la facultad 2: Madelín, Eddry, Glauver y Antonio.

A mis amistades de la Facultad 5: David Pino, Enrique, Jose Carlos.

Y aquellas personas que una forma u otra estuvieron presentes en mi paso por la UCI.

Declaración de autoría

Declaro ser la única autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas poseedora de los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Lázara Dalia Monteagudo Campos

Autora

_____

Ing. Ángel Ulise Tabares González

Tutor

_____

Ing. Karel Piorno Charchabal

Tutor

_____

Ing. Víctor Alejandro Roque Domínguez

Co-Tutor

Resumen

En el presente trabajo investigativo se desarrolló un componente para el cálculo de puntos de intersección de curvas en 2D para el proyecto AsiXMec 2.0 del centro de Entornos Interactivos 3D, Vertex, de la Universidad de las Ciencias Informáticas (UCI). Debido a las particularidades que presenta el contexto de desarrollo, se decidió utilizar XP como metodología de desarrollo de software y el framework Open CASCADE Technology para el modelado paramétrico.

El componente de intersección obtenido transforma las entidades línea, círculo y elipse de su forma paramétrica, tal y como las brinda el framework Open CASCADE, a su representación matricial como ***curva cónica***[1], ya sea degenerada o no degenerada, y luego calcula los puntos de intercepción entre ellas.

Palabras clave: puntos de intersección de curvas, modelado paramétrico, representación matricial, curva cónica.

Índice

Índice de Figuras

ÍNDICE DE TABLA

Introducción

En los últimos años se han producido avances significativos en el proceso de diseño industrial, área que se enfrenta a la necesidad de alcanzar altos niveles la calidad, disminuir los costes y acortar el tiempo de diseño y producción. “El diseño y desarrollo de nuevos productos o la modificación de los existentes se ha convertido en un elemento clave y fundamental para la mejora de la capacidad de innovación y competitividad de las empresas industriales. La incorporación de la computadora es "en la producción" el elemento puente que permite lograr la automatización integral de los procesos industriales” (Lazo, Junio de 2006).

En la actualidad los software profesionales basados en el diseño industrial tienden a integrase bajo un gran sistema CAD/CAE/CAM ((CAD) diseño asistida por computadoras, (CAE) ingeniería asistida por computadoras y fabricación asistida por computadoras (CAM)). Los sistemas CAD/CAE/CAM posibilitan el diseño, ingeniería y fabricación asistido por computadoras de productos industriales, revistiendo una gran importancia por el potencial que presentan para gestión de los procesos industriales.

Para un mayor entendimiento “el término CAD atiende prioritariamente aquellas tareas exclusivas del diseño, hace referencia a una herramienta software que, mediante el uso del ordenador permite: crear y modificar modelos en dos y tres dimensiones (2D o 3D). Mientras que el término CAE permite integrar las propiedades del diseño del modelo de la pieza, condiciones a las que está sometido y calcular cómo va a comportarse en la realidad. Asimismo el término CAM son sistemas informáticos que permiten fabricar las piezas en máquinas de Control Numérico por Ordenador, donde calcula las trayectorias de la herramienta para conseguir el mecanizado correcto” (Bonilla, 2003).

La Universidad de las Ciencias Informáticas, como puntera de nuestro país en la puesta en funcionamiento de las más modernas tecnologías en beneficio de la sociedad, ha incursionado en el uso de los sistemas CAD/CAE/CAM. En el proyecto Diseño y Simulación de Estructuras Mecánicas (DISEM), perteneciente al centro Entornos Interactivos 3D (Vertex) de la Facultad 4, se encarga de desarrollar una alternativa cubana con el propósito de en un futuro cercano satisfacer las necesidades de las industrias locales. Este software, llamado AsiXMec, se

enfoca en el área CAD, ya que facilita el trabajo manual aportando herramientas de dibujo bajo un soporte informático. Contiene un conjunto de funcionalidades que requieren del cálculo de los puntos de intersección entre dos o más curvas.

En la actualidad AsiXMec permite crear en el modelador geométrico las curvas línea, circunferencia, arco de circunferencia, elipse, arco de elipse y spline. Sin embargo los métodos utilizados para el cálculo de los puntos de intercepción fueron diseñados en un momento en que solo existían líneas, circunferencias y arcos de circunferencias. Por lo que al aplicar estos procedimientos en las entidades más recientes, ha arrojado resultados que en ocasiones son incorrectos o inexactos ya que no fueron diseñados cuando intervienen estas curvas.

Teniendo en cuenta la situación antes expuesta se plantea como **problema de la investigación:** ¿Cómo obtener los puntos de intersección entre dos curvas paramétricas de tipo cónicas para AsiXMec 2.0?

El problema está enmarcado en el **objeto de estudio**: algoritmos para el cálculo de los puntos de intersección de curvas. A partir del mismo, se delimita como **campo de acción**: algoritmos para el cálculo de los puntos de intersección de curvas paramétricas de tipo cónicas en 2D.

Para solucionar el problema planteado, se define como **objetivo general:** desarrollar un componente para obtener los puntos de intersección de curvas paramétricas de tipo cónicas en 2D para AsiXMec 2.0.

Se proponen las siguientes **tareas de investigación:**

- ✓ Investigación de los principales conceptos y definiciones asociados a las curvas paramétricas que se manejan en AsiXMec para fundamentar las bases teóricas de la investigación.
- ✓ Selección del método para el cálculo de los puntos de intersección entre curvas paramétricas de tipo cónica.

- ✓ Selección de la metodología de desarrollo de software, herramientas y lenguaje de desarrollo empleados.

- ✓ Implementación del método seleccionado para interceptar curvas 2D.

- ✓ Validación del componente de intersección de curvas paramétricas en 2D para AsiXMec 2.0 para comprobar su correcto funcionamiento.

Como posibles resultados a alcanzar con el desarrollo de este trabajo están:

- ✓ Un componente para obtener los puntos de intersección de entidades 2D para el modelador geométrico de AsiXMec 2.0.

- ✓ La documentación técnica asociada al desarrollo de un componente en AsiXMec.

Para el desarrollo de la investigación se hizo uso de métodos científicos de investigación siendo estos la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. Se clasifican en teóricos y empíricos, los cuales están dialécticamente relacionados. (Hernández León, y otros, 2011)

Métodos teóricos: Permiten estudiar las características del objeto de estudio para identificar las relaciones que fluyen alrededor de este, contribuyendo al desarrollo de teorías científicas.

- ✓ **Analítico-Sintético:** Sirvió para realizar el procesamiento de toda la información, sintetizándola y diferenciándola para de esta forma enfocarla hacia el desarrollo del componente intersección.

Métodos empíricos: Representan un nivel del estudio que precede de la experiencia adquirida y se desarrolla una elaboración racional que describe y explica las características del fenómeno que se investiga.

Documental: Se utiliza para consultar bibliografía en fuentes de carácter documental tales como libros, artículos y ensayos.

El presente documento está compuesto por tres capítulos:

Capítulo I. Marco Teórico de la Investigación: En este capítulo se definen los principales conceptos asociados al dominio del problema que serán empleados durante todo el trabajo, se presentan las bases teóricas fundamentales relacionadas con la herramienta AsiXMec y Open CASCADE Framework. Se realiza la selección del método para el cálculo de puntos de intersección para curvas paramétrica de tipo cónica. Además se selecciona la metodología de desarrollo de software, herramientas y lenguaje de desarrollo empleados en el desarrollo del componente de intersección de curvas paramétricas en 2D para AsiXMec 2.0.

Capítulo II. Fases Planificación y Diseño: En este capítulo se precisan un conjunto de elementos para conformar la propuesta de solución. Se realiza una caracterización del algoritmo seleccionado y el comportamiento del componente de intersección. Se aplica la metodología de desarrollo seleccionada y se generan los artefactos correspondientes a las fases de exploración, planificación y diseño.

Capítulo III. Fases Desarrollo y Pruebas: En el presente capítulo se aborda la implementación del sistema perteneciente a la fase desarrollo. Donde se identifican y definen las tareas de ingeniería correspondientes. Además se abarca todo lo relacionado con las pruebas realizadas al componente, ya sean pruebas de unidad y aceptación, para validar que se hayan cumplido el objetivo propuesto.

Capítulo I. Marco Teórico de la Investigación

Introducción

En este capítulo se definen los principales conceptos asociados al dominio del problema que serán empleados durante todo el trabajo, se presentan las bases teóricas fundamentales relacionadas con la herramienta AsiXMec y OpenCascade Framework. Se realiza la selección del método para el cálculo de puntos de intersección para curvas paramétrica de tipo cónica. Además se selecciona la metodología de desarrollo de software, herramientas y lenguaje de desarrollo empleados en el desarrollo del componente de intersección de curvas paramétricas en 2D para AsiXMec 2.0.

1.1 Conceptos asociados al objeto de estudio del problema

1.1.1 Entidades

Es el término con el cual se define a las curvas: líneas, circunferencias, arcos de circunferencias, elipses, arcos de elipses que utiliza AsiXMec.

1.1.2 Entidades descritas según el framework Open CASCADE

Open CASCADE Technology describe cómo están relacionada las clases y su nivel de abstracción (ver Figura 1). A continuación se relacionan las entidades involucradas:

Geom_Curve: “La clase abstracta Geom_Curve describe el comportamiento común de las curvas en el espacio 3D. El paquete Geom proporciona numerosas clases concretas de curvas derivadas, incluyendo líneas, circunferencias, cónicas, curvas de Bézier o BSpline, etc. La característica principal de estas curvas es que están parametrizadas” (OPEN CASCADE S.A.S, 2015).

Geom_Line: Describe una línea infinita. Una línea se define y se coloca en el espacio con un eje que le da un origen y un vector unitario. La línea Geom_Line está parametrizada de la siguiente manera:

$$P(U) = O + U * Dir \qquad [1]$$

Donde:

- P es el punto de la curva correspondiente al parámetro U.
- O es el origen y Dir el vector unitario de su eje de posicionamiento.
- El rango de parámetros es $(-\infty, +\infty)$.

(OPEN CASCADE S.A.S, 2015)

Geom_Circle: Describe una circunferencia en el espacio 3D. Una circunferencia se define por su radio y, como con cualquier curva de tipo cónica, se sitúa en el espacio con un sistema de coordenadas para la mano derecha. La circunferencia Geom_Circle está parametrizada por:

$$P(U) = O + R * \mathrm{Cos}(U) * XDir + R * \mathrm{Sin}(U) * YDir \qquad [2]$$

Donde:

- P de la curva correspondiente al parámetro U.
- O, $XDir$ y $YDir$ son respectivamente el origen, la dirección en X y la dirección en Y de su sistema de coordenadas local.
- R es el radio de la circunferencia.

Una circunferencia es una curva cerrada y periódica. El periodo es 2π y el rango de los parámetros son de $[0, 2\pi)$ (OPEN CASCADE S.A.S, 2015).

Geom_Ellipse: Describe una elipse en el espacio del 3D. Está definida por su radio principal y radio menor y, al igual que con cualquier curva de tipo cónica, se sitúa en espacio con un sistema de coordenadas para la mano derecha. La elipse Geom_Ellipse es parametrizada por:

$$\mathrm{P}\,(\mathrm{U}) \;=\; \mathrm{O} \;+\; \mathrm{MajorRad} * \mathrm{Cos}\,(\mathrm{U}) * \mathrm{XDir} \;+\; \mathrm{MinorRad} * \mathrm{Sin}(\mathrm{U}) * \mathrm{YDir} \qquad [3]$$

Donde:

- P de la curva correspondiente al parámetro U,
- O, $XDir$ y $YDir$ son respectivamente el origen, la dirección en X y la dirección en Y de su sistema de coordenadas local,
- *MajorRad* y *MinorRad* son el radio principal y el radio menor de la elipse.

Una elipse es una curva cerrada y periódica. El período es 2π y el rango de los parámetros son de $[0, 2\pi)$ (OPEN CASCADE S.A.S, 2015).

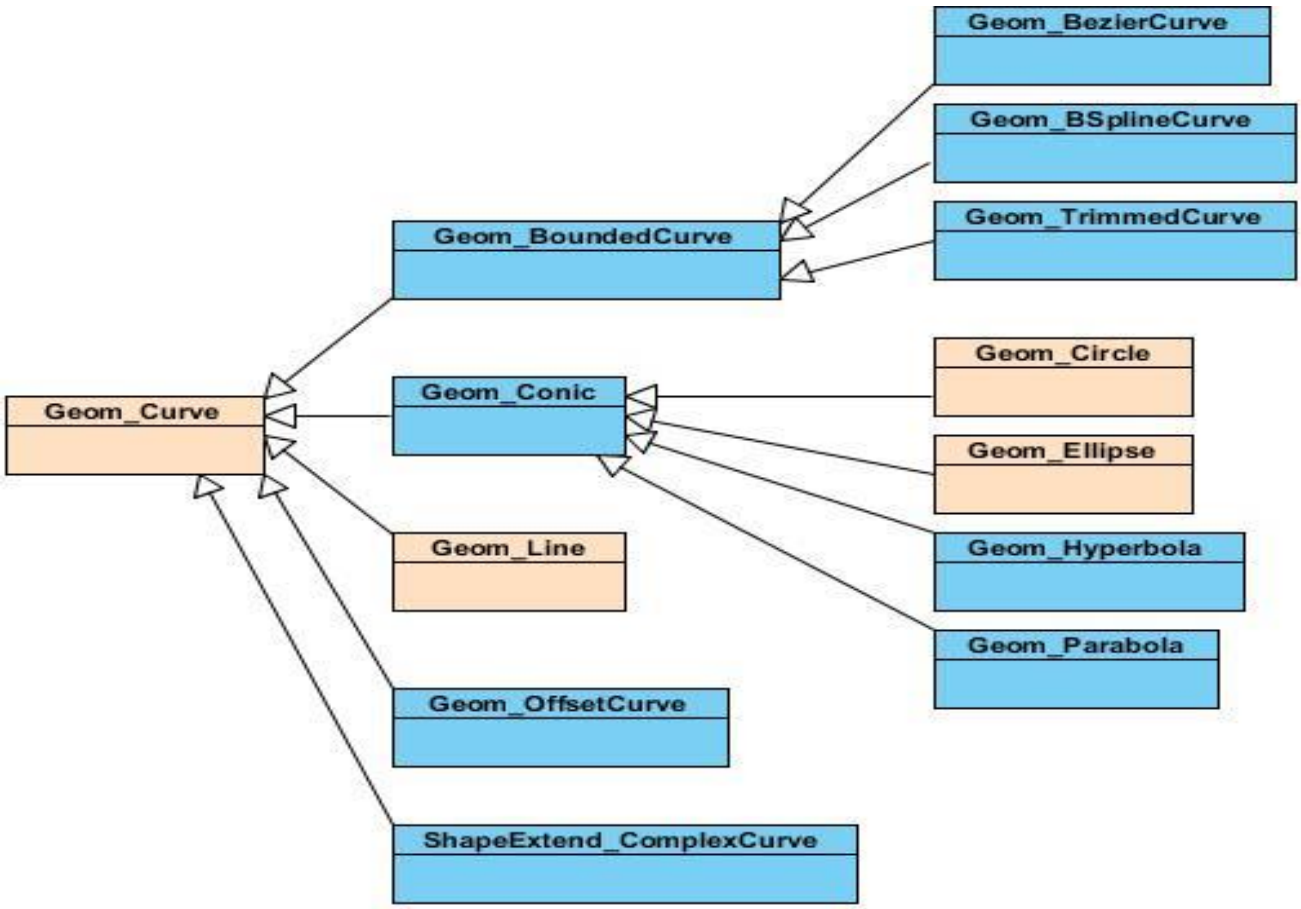

Figura 1 Diagrama de la jerarquía de clases del framework Open CASCADE para la representación de curvas. Fuente: (OPEN CASCADE S.A.S, 2015)

1.1.3 Representación de las entidades

Las entidades se pueden representar de diferentes formas matemáticamente. Las que se emplearán en este trabajo investigativo se relacionan a continuación:

Representación paramétrica de las entidades

"Se supone que x e y se definen como funciones de una tercera variable u, llamada parámetro, mediante las ecuaciones:

$$x = f(u)\ y = g(u)$$

Que se denominan ecuaciones parámetricas. Al variar u, el punto de $(x, y) = (f(u), g(u))$ cambia de posición y describe una curva C que llamamos curva paramétrica" (Stewart, 2008).

Representación matricial de las entidades de tipo cónica

Las ecuaciones de las cónicas son un caso particular de las ecuaciones cuadráticas que se da cuando la matriz de coeficientes es simétrica. Se utiliza los coeficientes de la ecuación de las entidades de tipo cónica para representar la matriz de la entidad, partiendo de la ecuación general de la cónica:

$$\mathrm{Ax}^2 + 2\mathrm{Bxy} + \mathrm{Cy}^2 + 2\mathrm{Dx} + 2\mathrm{Ey} + \mathrm{F} = 0 \qquad [6]$$

Su representación matricial es:

$$(x \quad y \quad 1)M\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0 \qquad [7]$$

$$M = \begin{pmatrix} A & C/2.0 & \mathrm{D}/2.0 \\ C/2.0 & B & E/2.0 \\ D/2.0 & E/2.0 & F \end{pmatrix}$$

(Universidad de Coruña, 2015)

La representación matricial de la **línea** según las ecuaciones [6] y [7]:

$$Dx + Ey + F = 0 \qquad [8]$$

$$M_L = \begin{pmatrix} 0 & 0 & \mathrm{D}/2.0 \\ 0 & 0 & E/2.0 \\ D/2.0 & E/2.0 & F \end{pmatrix} \qquad [9]$$

La representación matricial de la circunferencia según las ecuaciones [6] y [7]:

$$Ax^2 + y^2 + 2Dx + 2Ey + F = 0 \qquad [10]$$

$$M_L = \begin{pmatrix} A & 1.0 & \mathrm{D}/2.0 \\ 1.0 & 0 & E/2.0 \\ D/2.0 & E/2.0 & F \end{pmatrix} \qquad [11]$$

La representación matricial de la elipse según las ecuaciones [6] y [7]:

$$Ax^2 + Bxy + Cy^2 + 2Dx + 2Ey + F = 0 \; [12]$$

$$M_L = \begin{pmatrix} A & C & \mathrm{D}/2.0 \\ C & B/2.0 & E/2.0 \\ D/2.0 & E/2.0 & F \end{pmatrix} \quad [13]$$

1.1.4 Puntos de Intersección

"Considere dos ecuaciones independientes:

$$\mathrm{f(x,y)} = 0 \; [14] \; \mathrm{g(x,y)} = 0 \; [15]$$

Si sus gráficas se cortan en uno o más puntos, cada uno de estos puntos se llama punto de intersección. Como un punto de intersección de dos curvas $[14]$ y $[15]$ está sobre cada una de dichas curvas, sus coordenadas deben satisfacer, simultáneamente, ambas ecuaciones" (Lehmann).

1.2 Selección del método para el cálculo de los puntos de intersección

Para hallar los puntos de intersección existen varios métodos que resuelven mediante algoritmos las interrogantes siguientes: ¿cómo calcular los puntos de intersección? y ¿cómo calcular los puntos de intersección de entidades de tipo cónicas? Existen varias formas de obtener los puntos de intercepción de dos secciones cónicas, a continuación se relacionan algunas de forma general:

- Se sustituye una de las ecuaciones en la otra, siempre que esto sea posible, y luego se aplica alguno de los siguientes métodos: Método de Bisección, del Método de Regula-Falsi o del Método de Newton-Raphson para hallar los puntos de intersección.
- Se encuentran las intersecciones de las entidades cónicas mediante su representación matricial.

1.2.1 Método de Bisección

Para la resolución de la ecuación $f(x) = 0$ se basa en el Teorema de Bolzano que asegura la existencia de, al menos, una raíz de una función $f(x)$ en un cierto intervalo $[a, b]$, bajo ciertas condiciones.

Teorema de Bolzano: "Sea f: $[a, b] \rightarrow Real$ una función continua en $[a, b]$ tal que $f(a)\, f(b) < 0$. Entonces existe $c \in (a, b)$ tal que $f(c) = 0$" (Pasadas Fernández, 2017).

Suponiendo que $f(x)$ es continua y cambia de signo en los extremos de $[a, b]$. Basándose en el anterior teorema, podemos aproximar una solución de la ecuación $f(x) = 0$ dividiendo el intervalo inicial en dos iguales y eligiendo aquel en el que $f(x)$ cambia de signo. Después se repite el proceso hasta que se verifique algún criterio de parada.

Para garantizar que el error del Método de Bisección sea menor o igual que un cierto valor de tolerancia ε se aplica el siguiente resultado, donde n es la cantidad de iteraciones (Pasadas Fernández, 2017):

$$e_n = \frac{b-a}{2^{n+1}}$$

1.2.2 Método de Regula-Falsi

Se trata de realizar un refinamiento del Método de Bisección, eligiendo la aproximación m a distancias de a y b proporcionales a $f(a)$ y $f(b)$. La ecuación de la recta que pasa por los puntos $(a, f(a))$ y $(b, f(b))$ es:

$$\frac{y - f(a)}{f(b) - f(a)} = \frac{x - 1}{b - a}$$

De donde se tiene que el corte con el eje OX es, haciendo $x = 0$ y despejando y, el valor:

$$m = \frac{y - f(a)}{f(b) - f(a)}$$

Se verifica que:

- m_n converge más rápido que en el Método de Bisección.
- Un extremo es fijo.
- La amplitud de los intervalos no tiende a cero.
- No admite acotación del error (Pasadas Fernández, 2017).

1.2.3 Método de Newton-Raphson

Para encontrar las raíces (ceros) de una ecuación, necesitas encontrar la derivada de esa ecuación. En cada iteración n, considerar la recta tangente a $f(x)$ en $(x_n, f(x_n))$ y tomar como siguiente aproximación $x_n + 1$ la intersección de dicha tangente con el eje de abscisas. Por tanto, teniendo en cuenta que la ecuación de la recta tangente a la gráfica de $f(x)$ en el punto $(x_n, f(x_n))$ es:

$$y - f(x_n) = f'(x_n)(x - x_n)$$

Se tiene que:

$$x_{n-1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Observaciones sobre el método:

- Es más rápido.
- Requiere que $f'(x) \neq 0$.
- Elegir $x_0$puede ser delicado (Pasadas Fernández, 2017).

1.2.4 Método de intersección de cónicas utilizando representación matricial

Dos cónicas pueden poseer ninguno, dos o cuatro posiblemente coincidentes puntos de intercepción. Un método eficiente para encontrar estas soluciones explota la representación matricial homogénea de las secciones cónicas, o sea matrices de 3x3 simétricas que dependen de 6 parámetros.

El procedimiento para encontrar los puntos de intercepción consta de los siguientes pasos, donde las cónicas son representadas por matrices:

I. Dadas dos cónicas C_1 y C_2, considere el $lápiz\ de\ cónicas^2$ dado por su combinación lineal $\lambda C_1 + \mu C_2$.
II. Identificar los parámetros homogéneos (λ, μ) que corresponden a las cónicas degeneradas del lápiz. Mediante la condición de que $\det(\lambda C_1 + \mu C_2) = 0$, donde $\lambda = 1$ y se hallan los valores de μ dando solución a la ecuación de tercer grado que se genera.
III. Dada la cónica degenerada $C_0 = \lambda C_1 + \mu C_2$, identificar las dos, posiblemente coincidentes, líneas que la constituyen.
IV. Interceptar cada línea identificada con cualquiera de las dos cónicas originales, se puede hacer eficientemente usando la representación dual de la cónica C_0.
V. Los puntos de intercepción representarán las soluciones al sistema de ecuaciones original.

Nota: Este procedimiento está sostenido por varios fundamentos matemáticos. El primero es que debido a las propiedades simétricas de las representaciones matriciales de las cónicas, la combinación lineal de dos o más cónicas es una cónica. En segundo es que si el determinante de una matriz que representa una cónica es 0, entonces la cónica es degenerada (Stack Exchange Inc, 2013).

A criterio del autor de este trabajo se selecciona el método de intersección de cónicas utilizando su representación matricial. Este método tiene la ventaja de que en teoría debe arrojar valores exactos, aunque en la práctica no es posible debido a la aritmética finita del ordenador.

1.3 Herramientas de desarrollo

Las herramientas empleadas durante el desarrollo del componente se relacionan a continuación:

1.3.1 AsiXMec

“El software en su versión 2.0 es desarrollado por el proyecto DISEM del Centro Vertex, este es un modelador geométrico y paramétrico basado en Feature e historial de operaciones, permite a los usuarios manipular, crear, editar, importar y exportar archivos *.step*, *.iges*, *.stl*, compatibles con otras aplicaciones para Linux como Salome-Meca, DraftSight, LibreCAD y para Microsoft Windows como *AutoCAD*, *SolidWorks* e *Inventor*, entre otros de los más conocidos”. (González Olivera, 2016)

“Se ha desarrollado con lenguaje C++ para el desarrollo de la línea CAD-CAE, utiliza framework Open CASCADE Technology, framework Qt-5 y soporte para plugins. Cuenta con un solver de restricciones que utiliza la biblioteca Eigen C++ Library como biblioteca de algebra lineal, permite el diseño asistido por snap e identificación automática de caras. Su interfaz gráfica tiene soporte multilenguaje” (González Olivera, 2016).

1.3.2 Qt Creator 3.0.1 y Qt framework 5

Es un IDE multiplataforma que se ajusta a las necesidades de los desarrolladores Qt. QtCreator se centra en proporcionar características que ayudan a los nuevos usuarios de Qt a aprender y comenzar a desarrollar rápidamente, también aumenta la productividad de los desarrolladores. “El soporte de Qt para diferentes plataformas Linux es extenso y maduro. Los instaladores de Qt para Linux suponen que el sistema operativo del host proporciona un compilador, depurador y otras herramientas de desarrollo de C ++. Además, la creación de aplicaciones gráficas Qt requiere la instalación de bibliotecas y encabezados OpenGL” (The QT Company Ltd., 2016).

1.3.3 OPEN CASCADE Technology 6.9.0

“Open CASCADE Technology es una plataforma de desarrollo de software que ofrece servicios de modelado 3D de superficies y sólidos, intercambio de datos CAD y visualización. La mayor parte de la funcionalidad Open CASCADE está disponible en forma de bibliotecas C ++. Se puede aplicar mejor en el desarrollo de software que se ocupa de modelado 3D (CAD), fabricación / medición (CAM) o simulación numérica (CAE). Es software libre; puede redistribuirlo y / o modificarlo bajo los términos de la licencia GNU Lesser General Public

License (LGPL) versión 2.1, con excepción adicional. Está diseñado para ser altamente portátil y es conocido por trabajar en una amplia gama de plataformas (UNIX, Linux, Windows, Mac OS X, Android)" (OPEN CASCADE S.A.S, 2015).

1.3.4 Visual Paradigm

Visual Paradigm para UML es una herramienta para desarrollo de aplicaciones utilizando modelado UML ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Visual Paradigm también ofrece:

- Navegación intuitiva entre la escritura del código y su visualización;
- Potente generador de informes en formato PDF/HTML;
- Documentación automática Ad-hoc;
- Ambiente visualmente superior de modelado;
- Sofisticado diagramador automáticamente de layout;

Visual Paradigm for UML Enterprise Edition (VP-UML EE): Es la edición top de la línea de productos, lo que representa todo lo más moderno y agrega valor en términos de modelado de datos orientado a objetos, hace posible la documentación del proyecto, mapeo relacional de objetos para Java, .NET y PHP, reduciendo costos y aumentando su productividad (González Olivera, 2016).

1.3.1 Eigen Library

"Eigen es una biblioteca de plantillas C ++ para álgebra lineal: matrices, vectores, solucionadores numéricos y algoritmos relacionados. Soporta todos los tamaños de matrices, desde pequeñas matrices de tamaño fijo hasta matrices densa arbitrariamente grandes, e incluso matrices escasas. La API es extremadamente limpia y expresiva mientras se siente natural para los programadores de C++, gracias a las plantillas de expresión. Es fiable ya que es probado exhaustivamente a través de su propia suite de pruebas (más de 500 ejecutables), la suite de pruebas BLAS estándar y partes de la suite de pruebas LAPACK. Tiene un buen soporte para el compilador mientras que se ejecuta el conjunto de

pruebas contra muchos compiladores para garantizar la confiabilidad y trabajar alrededor de cualquier error del compilador" (Benoît Jacob, 2008).

1.3.2 Boost Test Library Unit Test

Proporciona una solución fácil de usar y flexible para este dominio de problema: implementación y organización de la prueba de unidad C++. Realiza un seguimiento de todas las afirmaciones de las herramientas de prueba pasadas / fallidas. Comprueba el progreso de la prueba y genera un informe de resultados en varios formatos diferentes. Suministra corredores de prueba de línea de comandos que inicializan el marco y ejecutan las pruebas solicitadas (Rozental, 2007).

1.4 Lenguajes de desarrollo

Como lenguajes de programación se utiliza:

1.4.1 UML

"Unified Modeling Language o Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos y describe la semántica esencial de estos diagramas y los símbolos en ellos utilizados. Este lenguaje resultó de gran utilidad durante el desarrollo del módulo, pues con él se realizó el diseño de los diagramas, específicamente en el diseño del modelo conceptual y el diagrama de casos de uso del sistema" (Nuñez, 2016).

1.4.2 C++

"Es uno de los lenguajes de programación más populares y más utilizados, especialmente cuando se trata de desarrollo de software profesional. Su funcionalidad permite un uso mucho más amplio para todo tipo de programas / aplicaciones que la mayoría de los lenguajes de programación más sencillos para principiantes. Se puede utilizar en la mayoría de cualquier tipo de plataforma. Soporta la programación orientada a objetos" (Markus Dieterle, Alternative-Computer-Programming.com, 2016).

“Es un lenguaje imperativo orientado a objetos derivado del C. Posee características tales como portabilidad, brevedad, programación modular, velocidad y compatibilidad con Open CASCADE Technology 6.9.0” (Sierra, 2012).

1.5 Metodologías de desarrollo de software

Todo proceso de desarrollo de software es riesgoso y difícil de controlar. Si no se sigue una metodología para controlar el flujo de trabajo, probablemente se obtendrán clientes insatisfechos con el resultado. Para seleccionar metodología de desarrollo de software se usa el método de la estrella de Boehm-Turner. El cual plantea cinco criterios fundamentales mediante los que se valora el proyecto:

- Tamaño del equipo: representa el número de personas involucradas en el proyecto. Teniendo en cuenta el nivel de complejidad en la comunicación y los costos que pueden provocar cambios esperados.
- Criticidad del producto: permite evaluar la naturaleza del daño ocasionado por defectos que no hayan sido detectados al producto. Su evaluación puede ser cualitativa.
- Dinamismo de los cambios: representa la rapidez con la que pueden cambiar los requisitos del proyecto.
- Personal con que se cuenta: representa la proporción del personal con experiencia alta, media y baja.
- Cultura del equipo: la vinculación entre las organizaciones y las personas del proyecto pueden depender de la confianza o de la relación contractual. Esto reflejando el nivel de ceremonia necesario y aceptado: documentación, control, formalismo en las comunicaciones.

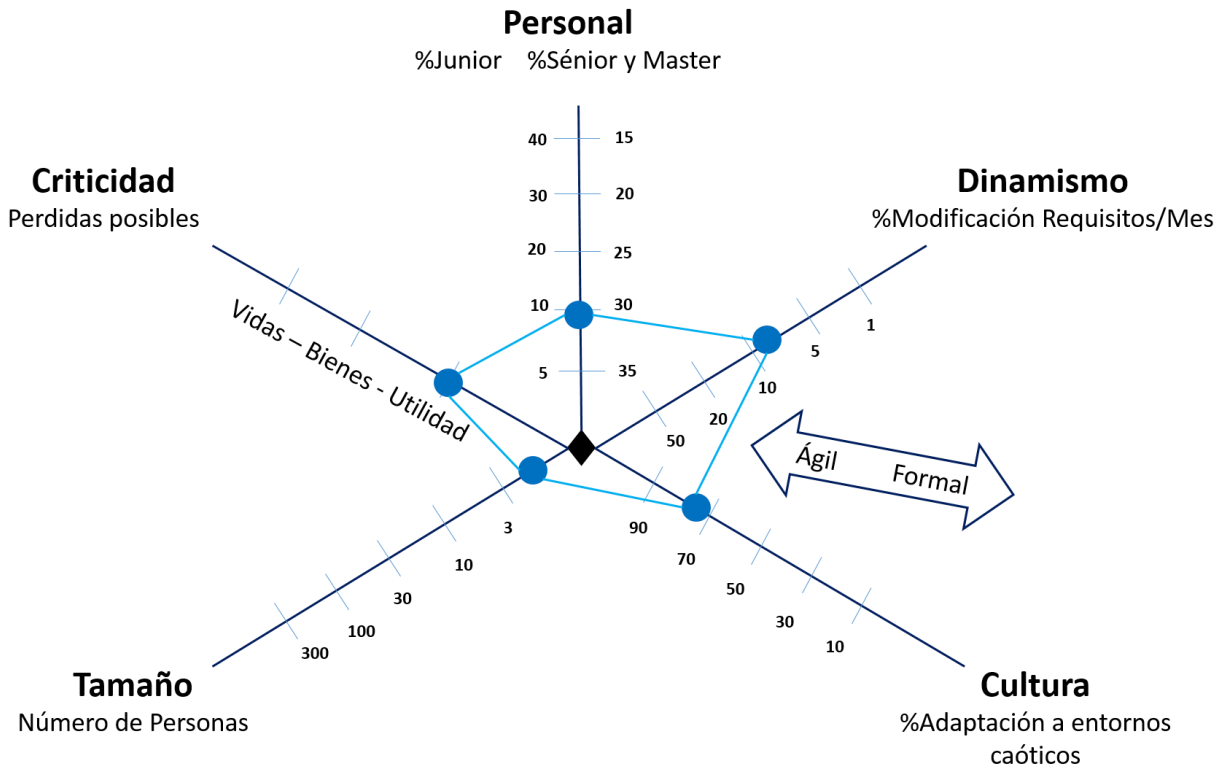

Figura 2 Método de la estrella de Boehm-Turner. Fuente: Elaboración propia

El método de la estrella de Boehm Turner arrojó según las variables de dinamismo, cultura, tamaño, criticidad y personal el tipo de metodología más adecuada a aplicar es la metodología ágil (Conferencia: Enfoques de IS. Modelos y metodologías del PDS, 2016).

1.5.1 Selección del método de desarrollo ágil

Entre los principales métodos ágiles tenemos el XP (eXtreme Programming) y Scrum. Los cuales tienen características propias, relacionadas a continuación:

Scrum:

“En Scrum un proyecto se ejecuta en bloques temporales (iteraciones-sprints). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea susceptible de ser entregado con el mínimo esfuerzo cuando el cliente lo solicite. El Sprint es el ritmo de los ciclos de Scrum. Está delimitado por la reunión de planificación del sprint y la reunión

retrospectiva. Al final del sprint se entrega el producto al cliente en el que se incluye un incremento de la funcionalidad que tenía al inicio del sprint" (María José Pérez Pérez, 2012).

Figura 3 Ciclo de Scrum – Sprint. Fuente: (María José Pérez Pérez, 2012)

XP:

"Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software. Promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes. Simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos

imprecisos y muy cambiantes, y donde existe un alto riesgo técnico” (María José Pérez Pérez, 2012).

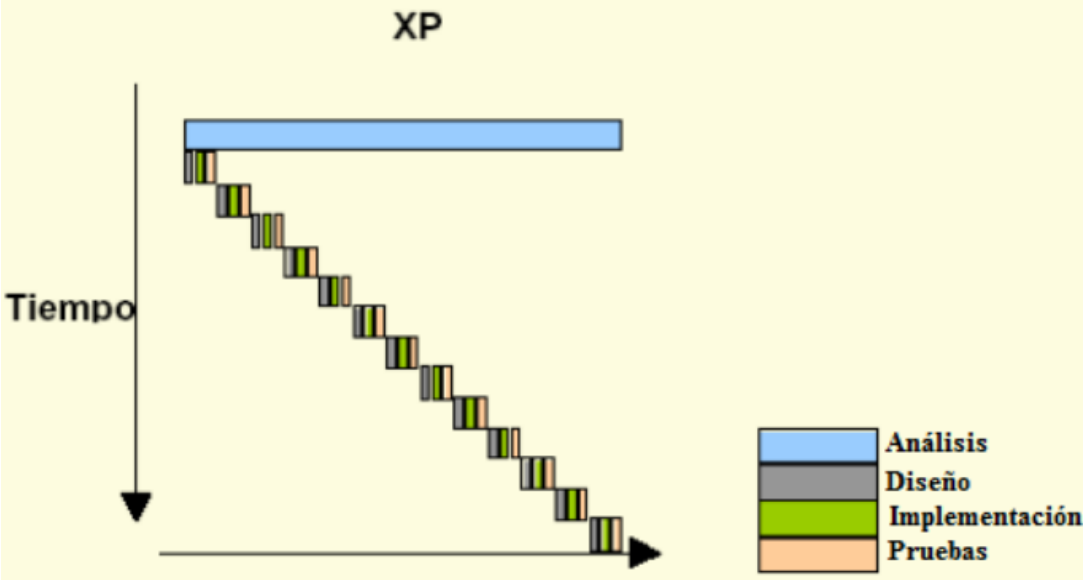

Figura 4 Ciclo de vida de XP. Fuente: (María José Pérez Pérez, 2012)

Las metodologías ágiles de desarrollo XP y SCRUM comparten características similares: el tamaño del proyecto deben ser pequeño o mediano y tamaño del equipo pequeño. Sin embargo debido a que SCRUM está centrada en la gestión de proyectos y XP por su parte se centra en la programación y obtención de un producto final, además genera documentación del software desarrollado. Se selecciona esta última como metodología de desarrollo para la presente investigación a través de las fases: exploración, planificación, diseño, desarrollo y pruebas.

1.5.2 Características principales de XP

- Se basa en realimentación continua entre el cliente y el equipo de desarrollo (prueba y error).
- Está orientada hacia quien produce y usa el software.
- Simplicidad en las soluciones implementadas
- Programación organizada.
- Satisfacción del programador.

Consideraciones del capítulo

Luego de desarrollar el capítulo se ha podido arribar a las siguientes conclusiones parciales:

- ✓ El análisis de los principales conceptos asociados al dominio del problema permitió tener una mejor comprensión de la solución informática a desarrollar.
- ✓ El análisis de los algoritmos para el desarrollo del componente, permitió corroborar que cumplen en su totalidad con las necesidades de la solución propuesta.
- ✓ El estudio de las principales herramientas, tecnologías y metodologías permitió seleccionar las más adecuadas para el desarrollo del sistema.

Capítulo II. Fases Planificación y Diseño

Introducción

En este capítulo se precisan un conjunto de elementos para conformar la propuesta de solución. Se realiza una caracterización del algoritmo seleccionado y el comportamiento del componente de intersección. Se aplica la metodología de desarrollo seleccionada y se generan los artefactos correspondientes a las fases de exploración, planificación y diseño.

2.1 Descripción de la propuesta de solución

La propuesta de solución es un componente de intersección entre curvas 2D para AsiXMec 2.0, el cual permite hallar los puntos de intersección de las entidades paramétricas de tipo cónica representadas. Se propone como estructura del componente dos clases: GeomTool e Intersect, donde la primera será la interfaz entre él y el módulo Cad-Core. La segunda clase está presente la transformación de entidades parametrizadas a la representación matricial y el algoritmo de intersección de las entidades parametrizadas de tipo cónica (Figura 5).

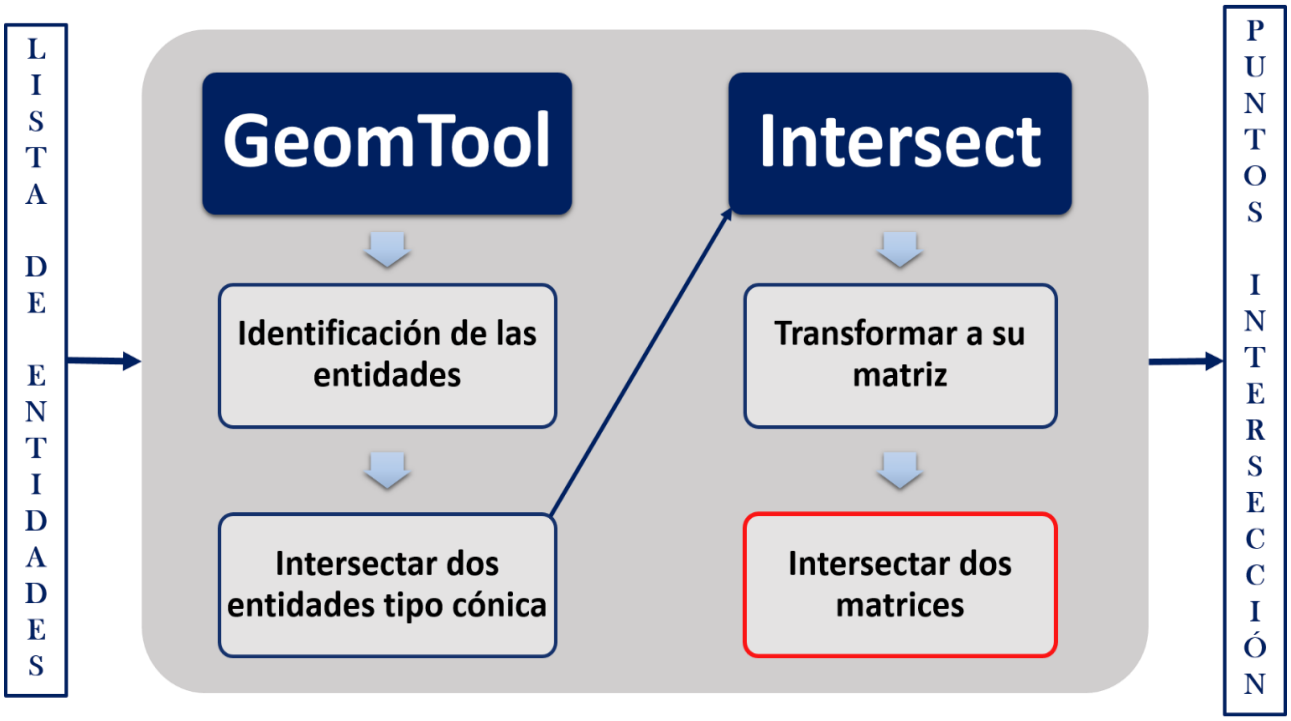

Figura 5 Componente de Intersección. Fuente: Elaboración propia

2.1.1 Transformación de la entidad parametrizada a su representación matricial

Anteriormente se ha señalado que las entidades están parametrizadas, por tanto se transformará a la ecuación general de una cónica utilizado las ecuaciones de la [1] a la [3] y de [6] a la [13]:

Transformación de la **línea** a su representación matricial (ecuaciones [1], [8] y [9]):

$x \equiv posición\ de\ O\ en\ eje\ "X"$ $y \equiv posición\ de\ O\ en\ eje\ "Y"$

$$A = 0, B = 0, C = 0, D = \tan(U), \mathrm{E} = -1.0, \mathrm{F} = (-\mathrm{D}) * \mathrm{x} + \mathrm{y}$$

$$M_L = \begin{pmatrix} 0 & 0 & \mathrm{D}/2.0 \\ 0 & 0 & E/2.0 \\ D/2.0 & E/2.0 & F \end{pmatrix}$$

Transformación de la **circunferencia** a su representación matricial (ecuaciones [2], [10] y [11]):

$x \equiv posición\ de\ O\ en\ eje\ "X"$ $y \equiv posición\ de\ O\ en\ eje\ "Y"$ $r \equiv radio$

$$A = 1.0, B = 0, C = 1.0, D = -2.0 * \mathrm{x}, \mathrm{E} = -2.0 * \mathrm{y}, \mathrm{F} = \mathrm{x} * \mathrm{x} + \mathrm{y} * \mathrm{y} - \mathrm{r} * \mathrm{r}$$

$$M_c = \begin{pmatrix} A & 1.0 & \mathrm{D}/2.0 \\ 1.0 & 0 & E/2.0 \\ D/2.0 & E/2.0 & F \end{pmatrix}$$

Transformación de la **elipse** a su representación matricial (ecuaciones [3], [12] y [13]):

$a \equiv radio\ mayor \qquad b \equiv radio\ menor$

$x \equiv posición\ de\ O\ en\ eje\ "X" \qquad y \equiv posición\ de\ O\ en\ eje\ "Y"$

$$A = \mathrm{a} * \mathrm{a} * (\sin(\mathrm{t}) * \sin(\mathrm{t})) + \mathrm{b} * \mathrm{b} * (\cos(\mathrm{t}) * \cos(\mathrm{t})),$$

$$B = 2.0 * (\mathrm{b} * \mathrm{b} - \mathrm{a} * \mathrm{a}) * \sin(\mathrm{t}) * \cos(\mathrm{t}),$$

$$C = \mathrm{a} * \mathrm{a} * (\cos(\mathrm{t}) * \cos(\mathrm{t})) + \mathrm{b} * \mathrm{b} * (\sin(\mathrm{t}) * \sin(\mathrm{t}))$$

$$D = -2.0 * \mathrm{A} * \mathrm{x} - \mathrm{B} * \mathrm{y}, \quad \mathrm{E} = -\mathrm{B} * \mathrm{x} - 2.0 * \mathrm{C} * \mathrm{y},$$

$$F = \mathrm{A} * \mathrm{x} * \mathrm{x} + \mathrm{B} * \mathrm{x} * \mathrm{y} + \mathrm{C} * \mathrm{y} * \mathrm{y} - \mathrm{a} * \mathrm{a} * \mathrm{b} * \mathrm{b}$$

$$M_e = \begin{pmatrix} A & C & \mathrm{D}/2.0 \\ C & B/2.0 & E/2.0 \\ D/2.0 & E/2.0 & F \end{pmatrix}$$

2.1.1 Intersección de cónicas utilizando la representación matricial

Para intersectar dos entidades de tipo cónica se tendrán las matrices $E_1\ y\ E_2$, donde se utiliza el algoritmo que se ejecuta como se muestra la tabla 1.

Tabla 1 Seudocódigo del algoritmo de Intersección cónica usando representación matricial

Algoritmo	Intersección cónica usando representación matricial
Entradas	Matrices que representan secciones cónicas: E_1 y E_2
Salidas	El conjunto P de los puntos de intersección de las entidades de tipo cónicas
Descripción	Permite hallar los puntos de intersección entre dos matrices de entidades de tipo cónica, mediante operaciones de algebra lineal.

Inicio

1. Comprobar que $\boldsymbol{E_1}$ y $\boldsymbol{E_2}$ sean **degenerada** o **no degeneradas**
2. **Si** $\boldsymbol{E_1}$ y $\boldsymbol{E_2}$ son **no degenerada**

   2.1 Hallar raíces $(\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3})$ del polinomio característico $\left|\boldsymbol{E_1} * (-\boldsymbol{E_2})^{-1}\right|$

   2.2 Tomar $\boldsymbol{r_i} \in \mathbb{R}$ entonces

   2.2.1 Obtener la degenerada $\boldsymbol{E_0} \leftarrow \boldsymbol{E_1} + \boldsymbol{r_i} * \boldsymbol{E_2}$

   2.2.2 Descomponer la degenerada en rectas $\boldsymbol{m}$ y $\boldsymbol{l}$

   Mientras $\mathbf{m} = \mathbf{0}$ y $\exists\ \boldsymbol{r_i} \in \mathbb{R}$

   2.3 Si $\mathbf{m} = \mathbf{0}$

   La solución es $\mathbf{P} \leftarrow \emptyset$

   2.4 Sino

   2.4.1 Nombrar $\boldsymbol{E_c}$ **a la cónica** $\boldsymbol{E_1}$

   2.4.2 Se halla las intersecciones $\boldsymbol{P_1}$ entre $\boldsymbol{E_c}$ y $\boldsymbol{m}$

   2.4.3 Se halla las intersecciones $\boldsymbol{P_2}$ entre $\boldsymbol{E_c}$ y $\boldsymbol{l}$

   2.4.4 La solución es $\mathbf{P} \leftarrow\ \mathbf{P_1} \cup \mathbf{P_2}$
3. **Sino** $\boldsymbol{E_1}$ o $\boldsymbol{E_2}$ son **degenerada, entonces**

   3.1 Se selecciona la matriz degenerada $\boldsymbol{E_d}$ y nombramos $\boldsymbol{E_c}$ a la otra cónica

   3.2 Se descompone $\boldsymbol{E_d}$ en las rectas $\boldsymbol{m}$ y $\boldsymbol{l}$

   3.3 Se hallan las intersecciones $\boldsymbol{P_1}$ entre $\boldsymbol{E_c}$ y $\boldsymbol{m}$

   3.4 Se hallan las intersecciones $\boldsymbol{P_2}$ entre $\boldsymbol{E_c}$ y $\boldsymbol{l}$

   3.5 La solución es $\mathbf{P} \leftarrow\ \mathbf{P_1} \cup \mathbf{P_2}$

FIN

Para una mejor visualización del algoritmo se muestra el diagrama de flujo correspondiente al seudocódigo anteriormente descrito (Figura 6).

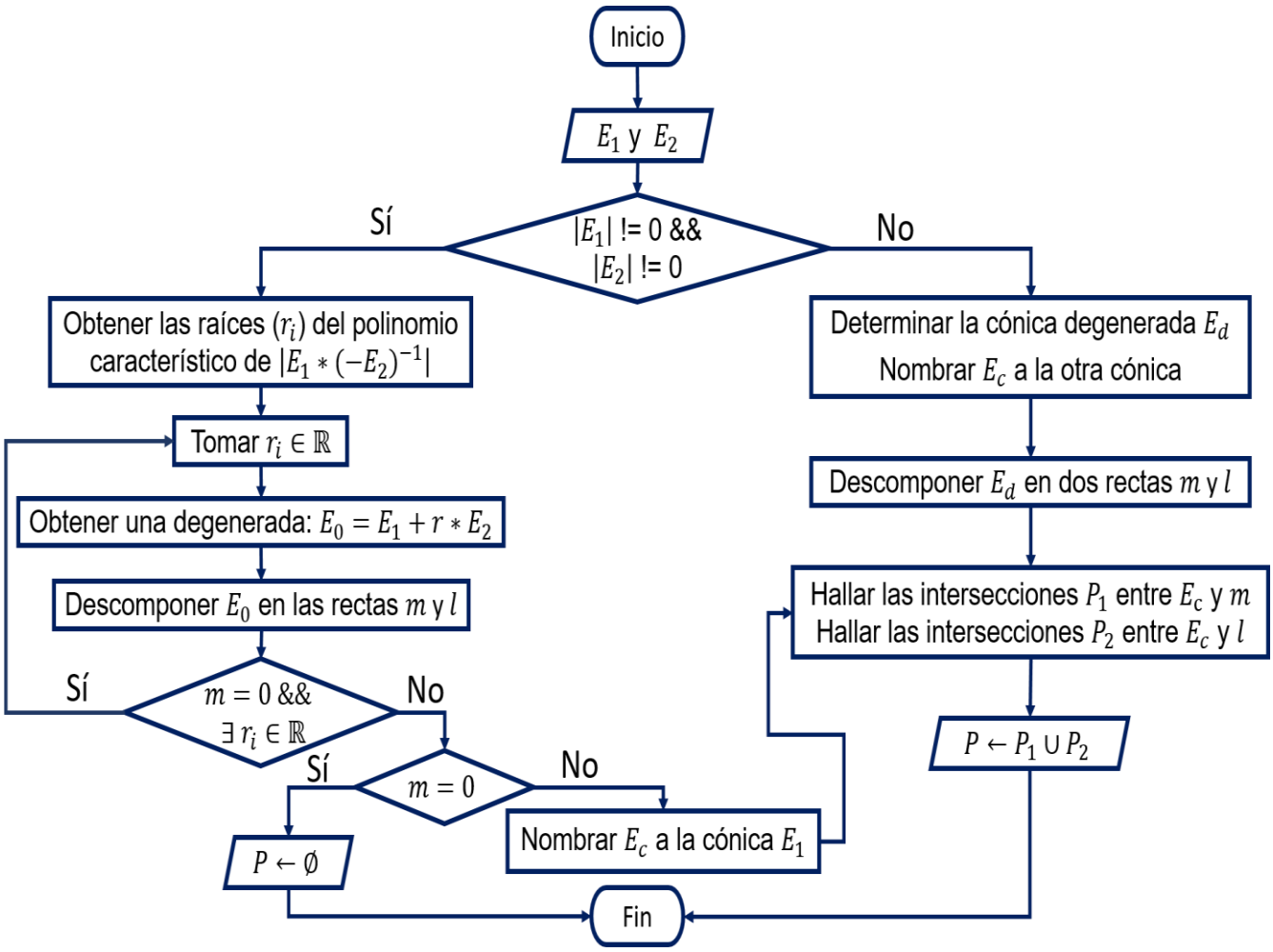

Figura 6 Intersección de cónicas utilizando la representación matricial. Fuente: Elaboración propia

2.1.2 Cónicas degenerada o no degenerada

Según sean los valores del determinante de la matriz asociada a la cónica E se tiene la siguiente clasificación:

- $|E| \neq 0 \equiv$ cónicas no degeneradas
- $|E| = 0 \equiv$ cónicas degeneradas

(Palencia González, y otros, 2016)

2.1.3 Descomponer matriz degenerada

Geométricamente, una cónica degenerada puede ser representada por:

- Un par de líneas rectas reales y un punto imaginario
- Un par de líneas rectas reales y un punto real (la intersección de las dos líneas)
- Un doble punto real y un par de líneas rectas imaginarias

Sea C una cónica degenerada de rango 1 está representada por una doble línea recta y tiene la forma:

$$C = l^T * l$$

- Para descomponer C en la recta l podemos simplemente recoger la fila i (y la columna j) de C relacionada con cualquiera de los elementos que no desaparecen Cij:

$$l \sim C_i$$

Ejemplo: Sea C_d una cónica degenerada de rango 1:

$$C_d = \begin{bmatrix} 2 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 2 \end{bmatrix}$$

Al seleccionar el elemento C_{d11} se puede recuperar la línea:

$$l = [2 \quad 0 \quad 2] \sim [1 \quad 0 \quad 1]$$

Notar que:

$$l^T * l = \begin{bmatrix} 4 & 0 & 4 \\ 0 & 0 & 0 \\ 4 & 0 & 4 \end{bmatrix} \sim C_d$$

En el caso de una matriz de rango 2 con valores propios reales, la cónica degenerada está compuesta por dos líneas intersectadas l y m y su punto de intersección p. Esta cónica tiene la forma:

$$C = l^T * m + m^T * l$$

Además, cualquier punto de una de las dos líneas l o m pertenece a la cónica:

Si $\boldsymbol{p}$ está en l entonces $l \cdot p = 0$ y:

$$p^T C p = p^T l^T * mp + p^T m^T * lp = 0 * mp + p^T m^T * 0 = 0$$

Para descomponer esta matriz se recupera el punto de intersección p y posteriormente la matriz $l^T \cdot m$ (Stack Exchange Inc, 2013).

2.1.4 Intersección de una cónica y una recta

Considerando una cónica dada por la matriz simétrica E . Sean $P = (p)$ y $Q = (q)$ los puntos de la recta. Se calcula en coordenadas homogéneas la intersección de la recta y la cónica:

$$recta\ PQ \equiv (x) = \alpha(p) + \beta(q)$$

$$cónica \equiv (x)\ E\ (x)^t = 0$$

Se sustituye la primera ecuación en la segunda:

$$\big(\alpha(p) + \beta(q)\big)E\big(\alpha(p) + \beta(q)\big)^t = 0 \leftrightarrow \alpha^2(p)E(p)^t + 2\alpha\beta(p)E(q)^t + \beta^2(q)E(q)^t = 0$$

Si $(p)E(p)^t = (p)E(q)^t = (q)E(q)^t = 0$ la ecuación se cumple para cualquier (α, β) luego la recta está contenida en la cónica.

En otro caso, se obtiene una ecuación de segundo grado de discriminante:

$$\frac{1}{4}\Delta = [(p)E(q)^t]^2 - [(p)E(p)^t][(q)E(q)^t]$$

Donde puede ocurrir una de tres posibilidades:

- $\Delta > 0$: Recta secante. Hay dos soluciones reales distintas, luego la recta corta a la cónica en dos puntos distintos.
- $\Delta = 0$: Recta tangente. Hay una solución doble, luego la recta corta a la cónica en un punto doble.
- $\Delta < 0$: Recta exterior. No hay soluciones reales, la recta no corta a la cónica (Universidad de Coruña, 2015).

2.2 Fase de Exploración

En esta fase los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Asimismo, el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizan en el proyecto.

2.2.1 Descripción de las Historias de Usuario (HU)

Las HU sustituyen a los documentos de especificación funcional, y a los “casos de uso”. Las principales características que presentan es que son independientes unas de otras, negociables, valoradas por los clientes o usuarios, estimables, pequeñas y verificables.

Para la confección de las HU se realizaron diferentes entrevistas a desarrolladores del módulo de Cad-Core del software AsiXMec2.0. Si bien el cliente no fue quien escribió personalmente las HU, fue él quien diseñó su contenido, asignó la prioridad de cada una de ellas y dirigió la redacción de las mismas, y además se encargó de asignarle una prioridad a cada HU. El equipo de desarrollo por su parte revisó esta prioridad analizando la dependencia entre HU y asignó el costo de cada una de ellas, este se traduce en las semanas para su desarrollo.

Las HU se representan mediante tablas las cuales contienen las siguientes secciones:

- **Código:** Las siglas de HU más un numero consecutivo, este permite la historia de usuario.
- **Nombre:** Nombre que identifica la HU.
- **Referencia:** Es el conjunto de códigos de las diferentes HU de las cuales depende actualmente la que se encuentra en desarrollo.
- **Prioridad:** Esta característica es dada por el cliente con los valores: alta, media o baja en dependencia de la importancia en que desean ser implementadas.
- **Iteración asignada:** Número de la iteración en la cual se desarrollara la HU.
- **Puntos estimados:** Tiempo estimado en semanas que se le asignara.
- **Descripción:** Breve descripción del proceso que define la HU.
- **Observaciones:** Alguna acotación importante de señalar acerca de la HU.

Tabla 2 HU1 Obtener los puntos de intersección entre varias entidades

HISTORIA DE USUARIO	
Código: HU1	**Nombre:** Intersectar dos o más entidades.
Responsable: L. Dalia Monteagudo	**Prioridad:** Alta
Iteración Asignada: 1	**Puntos estimados:** 3.0
Descripción: El usuario dibuja las entidades en la aplicación y el sistema devuelve en forma de punto, señalando estos puntos en las entidades.	
Observaciones: Hay funcionalidades del software que dependen los puntos de intersección.	

Tabla 3 HU2 Obtener los puntos de intersección entre dos entidades de tipo cónica

HISTORIA DE USUARIO	
Código: HU2	**Nombre:** Intersectar dos entidades de tipo cónica.
Responsable: L. Dalia Monteagudo	**Prioridad:** Alta
Iteración Asignada: 1	**Puntos estimados:** 3.0
Descripción: Dada dos entidades de tipo cónica se transforman a su representación matricial y se hallan los intersección entre dos matrices cónicas.	
Observaciones: Hay funcionalidades del software que dependen los puntos de intersección.	

Se confeccionaron un total de 2 HU, tomando para los puntos estimados la unidad como una semana de trabajo. Las HU 1 es la encargada comportamiento del componente y las HU 2 describe el método a emplearse para la solución del problema de investigación.

2.3 Fase de Planificación

2.3.1 Estimación del esfuerzo por HU

Las HU se priorizan de acuerdo el alcance de cada una de las entregas o release. El cliente decide las historias que se seleccionan para cada iteración. A continuación se presenta la tabla 5 donde se resume la estimación del esfuerzo realizada por parte de los desarrolladores y la duración total que tendrá el desarrollo del mecanismo de depuración como solución propuesta.

Tabla 4 Resumen de la estimación del esfuerzo

HISTORIAS DE USUARIO	PUNTOS DE ESTIMACIÓN (semanas)
HU1: Intersectar dos o más entidades.	**3.0**
HU2: Intersectar dos entidades de tipo cónicas	**3.0**
Total	**6.0**

2.3.2 Plan de Iteraciones

Luego de identificar y redactar cada una de las HU y de la estimación del esfuerzo necesario para realizarlas, se debe conformar el plan de iteraciones. Las HU seleccionadas para cada iteración son desarrolladas y probadas de acuerdo al orden preestablecido.

El desarrollo del componente de intersección fue dividido en 2 iteraciones teniendo en cuenta la complejidad de los métodos analizados y seleccionados posteriormente. A continuación se describen cada una de las iteraciones.

Iteración 1: En esta iteración se desarrollan las historias de usuario relacionadas la comunicación con los módulos de la aplicación.

Iteración 2: En esta iteración se desarrolla las historias de usuario relacionadas con la intersección entre entidades spline y línea. Se realiza una entrega funcional al finalizar la iteración en la cual se puede realizar el proceso de revisión y análisis de los resultados obtenidos.

Para aproximar el tiempo de ejecución de cada iteración, se tomó como medida que cada semana constaba de 5 días en los que se trabajaban 6 horas sin distracciones. En la tabla 6 se muestra el plan de iteraciones, este incorpora el tiempo estimado para cada una de las iteraciones y las HU que se van a desarrollar.

Tabla 5 Plan de iteraciones

ITERACIÓN	HISTORIA DE USUARIO	PUNTOS DE ESTIMACIÓN
1	**HU1:** Intersectar dos o más entidades.	**3.0**
2	**HU2:** Intersectar dos entidades de tipo cónicas	**3.0**

2.3.3 Plan de Entregas

A partir del plan de iteraciones analizado en el acápite anterior, y en correspondencia con el mismo se realiza el plan de entregas, que se muestra en la tabla 7. En el cual se proponen 2 versiones funcionales y una última entrega de iteraciones del producto el 1 de mayo, para dar paso a la fase de pruebas.

Tabla 6 Plan de entregas por iteración

HISTORIA DE USUARIO	1ra Iteración (23/04/2017)	2da Iteración (29/05/2017)
HU1: Intersectar dos o más entidades.	**V1.0**	
HU2: Intersectar dos entidades de tipo cónicas		**V2.0**

2.4 Fase de Diseño

La metodología XP sugiere que hay que conseguir diseños simples y sencillos, procurando hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible que a la larga costará menos tiempo y esfuerzo desarrollar.

2.4.1 Tarjetas clase-responsabilidad-colaboración

Las tarjetas clase-responsabilidad-colaboración (CRC) permiten desprenderse del método basado en procedimientos y trabajar con una metodología basada en objetos, así el programador se concentra y empieza a apreciar el desarrollo orientado a objetos. Las tarjetas CRC representan objetos y se describen partir de los siguientes elementos.

Clase: nombre de la clase a la cual pertenece la tarjeta.

Responsabilidad: describe cuales son las funcionalidades que deben ser implementadas por la clase.

Colaboración: enumera las diferentes clases con las cuales tiene relación la clase a la cual pertenece la tarjeta CRC.

Tabla 7 Tarjeta CRC GeomTool

TARJETA CRC	
Clase: GeomTool	
Responsabilidad: • intersect: intersección entre una entidad y una lista de entidades. • intersect: intersección entre dos entidades.	**Colaboración:** • clase: Intersect

Tabla 8 Tarjeta CRC Intersect

TARJETA CRC	
Clase: Intersect	
Responsabilidad: • intersect: intersección entre dos entidades de tipo línea	**Colaboración: -**

• intersect: intersección entre dos entidades de tipo línea y elipse. • intersect: intersección entre dos entidades de tipo circunferencia y circunferencia. • intersect: intersección entre dos entidades de tipo circunferencia y elipse.	

Consideraciones parciales

Luego de desarrollar el capítulo se ha podido arribar a las siguientes conclusiones parciales:

- ✓ El análisis de los algoritmos para el desarrollo del componente, permitió conformar la propuesta de solución de la presente investigación.
- ✓ El componente de intersección está compuesto por 2 historias de usuario, las cuales permitieron tener la base para la implementación de la aplicación.
- ✓ En la fase de planificación se determinó que la duración del desarrollo del componente intersección como solución propuesta es de 8 semanas, para lo cual contará con 2 iteraciones, al finalizar cada una se tendrá una entrega funcional de dicho componente.

Capítulo III. Fase de Desarrollo y Pruebas

Introducción

En el presente capítulo se aborda la implementación del sistema perteneciente a la fase desarrollo. Donde se identifican y definen las tareas de ingeniería correspondientes. Además se abarca todo lo relacionado con las pruebas realizadas al componente, ya sean pruebas de unidad y aceptación, para validar que se hayan cumplido el objetivo propuesto.

3.1 Tareas de ingeniería

Asociado a cada iteración se encuentra la planificación de las tareas de ingeniería o programación, cada HU se transforma en estas tareas que son desarrolladas por programadores, dentro del equipo de desarrollo, aplicando la práctica de la programación en parejas. Para cada iteración se realizó la distribución de tareas en correspondencia con las HU que se desarrollaron.

Tabla 9 Tareas de ingeniería

Historias de Usuario	Tareas de ingeniería
HU1: Obtener los puntos de intersección entre dos o más entidades.	1. Intersectar una entidad y una lista de entidades. 2. Intersectar dos entidades de tipo cónica.
HU2: Obtener los puntos de intersección entre dos entidades de tipo cónica.	1. Intersectar dos entidades de tipo línea-línea. 2. Intersectar dos entidades de tipo línea-circunferencia. 3. Intersectar dos entidades de tipo línea-elipse. 4. Intersectar dos entidades de tipo circunferencia-circunferencia. 5. Intersectar dos entidades de tipo circunferencia-elipse.

	6. Intersectar dos entidades de tipo elipse-elipse.

A continuación se presentan las tareas de ingeniería correspondiente a las historias de usuarios.

Tabla 10 Tarea de ingeniería HU1_T1

Tareas de ingeniería	
No. de tarea: 1	**No. de HU:** 1
Nombre de la tarea: Intersectar una entidad y una lista de entidades.	
Tipo de tarea: Desarrollo	**Responsable:** Lázara D. Monteagudo Campos
Descripción: Intersectar una entidad y una lista de entidades y hallar los puntos de intersección entre ellas.	

Tabla 11 Tareas de ingeniería HU1_T2

Tareas de ingeniería	
No. de tarea: 2	**No. de HU:** 1
Nombre de la tarea: Intersectar dos entidades de tipo cónicas.	
Tipo de tarea: Desarrollo	**Responsable:** Lázara D. Monteagudo Campos
Descripción: Se desarrolla la funcionalidad de tomar dos entidades y hallar los puntos de intersección.	

Tabla 12 Tareas de ingeniería HU2_T1

Tareas de ingeniería	
No. de tarea: 1	**No. de HU:** 2
Nombre de la tarea: Intersectar dos entidades de tipo línea-línea.	
Tipo de tarea: Desarrollo	**Responsable:** Lázara D. Monteagudo Campos
Descripción: Se desarrolla la funcionalidad para hallar los puntos de intersección entre dos líneas.	

Tabla 13 Tareas de ingeniería HU2_T2

Tareas de ingeniería	
No. de tarea: 2	**No. de HU:** 2
Nombre de la tarea: Intersectar dos entidades de tipo línea-circunferencia.	
Tipo de tarea: Desarrollo	**Responsable:** Lázara D. Monteagudo Campos
Descripción: Se desarrolla la funcionalidad para hallar los puntos de intersección entre línea-circunferencia.	

Tabla 14 Tarea de ingeniería HU2_T3

Tareas de ingeniería	
No. de tarea: 3	**No. de HU:** 2
Nombre de la tarea: Intersectar dos entidades de tipo línea-elipse.	
Tipo de tarea: Desarrollo	**Responsable:** Lázara D. Monteagudo Campos

Descripción: Se desarrolla la funcionalidad para hallar los puntos de intersección entre línea-elipse.

Tabla 15 Tarea de ingeniería HU2_T4

Tareas de ingeniería	
No. de tarea: 4	**No. de HU:** 2
Nombre de la tarea: Intersectar dos entidades de tipo circunferencia-circunferencia.	
Tipo de tarea: Desarrollo	**Responsable:** Lázara D. Monteagudo Campos
Descripción: Se desarrolla la funcionalidad para hallar los puntos de intersección entre circunferencia- circunferencia.	

Tabla 16 Tarea de ingeniería HU2_T5

Tareas de ingeniería	
No. de tarea: 5	**No. de HU: 2**
Nombre de la tarea: Intersectar dos entidades de tipo circunferencia-elipse.	
Tipo de tarea: Desarrollo	**Responsable:** Lázara D. Monteagudo Campos
Descripción: Se desarrolla la funcionalidad para hallar los puntos de intersección entre circunferencia- elipse.	

Tabla 17 Tarea de ingeniería HU2_T6

Tareas de ingeniería	
No. de tarea: 6	**No. de HU:** 2
Nombre de la tarea: Intersectar dos entidades de tipo elipse-elipse.	

Tipo de tarea: Desarrollo	**Responsable:** Lázara D. Monteagudo Campos
Descripción: Se desarrolla la funcionalidad para hallar los puntos de intersección entre elipse-elipse.	

3.2 Fase de pruebas

XP propone más énfasis en el proceso de pruebas que otros métodos ágiles. Las pruebas al sistema son fundamentales, ya que han desarrollado un enfoque que reduce la probabilidad de producir nuevos incrementos del sistema que introduzcan errores en el software existente. “Como se ha observado al transcurso de la investigación realizada: los requerimientos del usuario se expresan como escenarios o historias y el usuario establece las prioridades de estos para su desarrollo. El desarrollador evalúa cada escenario y lo divide en tareas, cada tarea representa una característica distinta del sistema y se puede diseñar entonces pruebas de unidad para esa tarea. El papel del cliente en el proceso de pruebas es diseñar las pruebas de aceptación para las historias que se tienen que implementar para la entrega del sistema” (Somerville).

3.2.1 Pruebas de Unidad

“La prueba de unidad es la disciplina de escribir código que prueba otro código. Las pruebas unitarias, que son código fuente, pueden ser compiladas y ejecutadas. Cuando se ejecuta cada prueba de unidad, reporta el éxito o el fracaso de la prueba con un simple indicador visual verdadero o falso, a menudo verde o rojo. Si todas las pruebas de unidad pasan, el código de producción que prueban se considera que posiblemente esté funcionando. Si incluso una sola prueba unitaria falla (de posiblemente miles) el código de producción en general se considera que sin duda se romperá” (McLean Hall, 2014).

Cada prueba unitaria se compone de tres partes distintas:

- La disposición de las condiciones previas de la prueba
- El desempeño del acto que se está probando
- La afirmación de que el comportamiento esperado ocurrió

Con el uso de la biblioteca Boost se realiza las pruebas unitarias al algoritmo de intersección de cónicas usando matrices. Donde se comprueba que dada las matrices cónicas se hallan sus raíces y se evalúan en la ecuación [7] (descrita en el primer capítulo) si se encuentran cerca de 0 y menor que 1. Mediante este procedimiento se obtuvo como resultados que no había ocurrencia de errores. Se realizan las siguientes pruebas unitarias a los métodos de intersección de cada entidad de tipo cónica:

- BOOST_AUTO_TEST_CASE(PruebaInterseccionLineaLinea)
- BOOST_AUTO_TEST_CASE(PruebaInterseccionLineaCirculo)
- BOOST_AUTO_TEST_CASE(PruebaInterseccionLineaElipse)
- BOOST_AUTO_TEST_CASE(PruebaInterseccionCirculoCirculo)
- BOOST_AUTO_TEST_CASE(PruebaInterseccionCirculoElipse)
- BOOST_AUTO_TEST_CASE(PruebaInterseccionElipseElipse)

Las cuales arrojaron resultados satisfactorios en cada caso, cumpliendo así el objetivo de las pruebas de unidad. A continuación se muestra un ejemplo al aplicar las pruebas unitarias utilizando la biblioteca de Boost:

```
BOOST_AUTO_TEST_CASE(PruebaInterseccionLineaLinea)
{
  std::cout << "PruebaInterseccionLineaLinea test" << std::endl;

  Intersect* cint = new Intersect();

    Matrix3d E1, E2;
    E1 << 0,0,1,0,0,-0.5,0,0.5,0;
    E2 << 0,0,0,0,0,-0.5,0,0.5,-2;

  std::vector<Eigen::Vector3d>  points = cint->intersectConics(E1,E2);

  BOOST_REQUIRE(points.size() == 2); //Verificar que exista los dos puntos de intersección

  for (auto p : points){
    double c1d = p.transpose() * E1 * p;
    double c2d = p.transpose() * E2 * p;
    BOOST_REQUIRE_SMALL(c1d, 1.0);//se garantiza que pertenezca a la 1era cónica
    BOOST_REQUIRE_SMALL(c2d, 1.0);//se garantiza que pertenezca a la 2da cónica
   }
}
```

Figura 7 PU1_InterseccionLíneaLínea. Fuente: Elaboración propia

3.2.2 Pruebas de Aceptación

“Las pruebas de aceptación son creadas en base a las HU, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que las HU han sido correctamente implementadas. Las pruebas de aceptación son consideradas como “pruebas de caja negra”, Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos” (Joskowicz, 2008).

Como criterio de aprobación de cada iteración se tomó que el 100% de los casos de prueba sean exitosos para pasar de iteración. El objetivo de estas pruebas no es tener un conjunto de casos escritos que cubran el 100% del código, sino poder realizarle pruebas al sistema desde el punto de vista del usuario.

Para representar las pruebas de aceptación se definieron los siguientes elementos:

- **Código:** Representa al caso de prueba, incluye el número de HU, de la prueba y si posee diferentes escenarios.
- **HU:** Número de la HU a la cual pertenece.
- **Nombre:** Junto al código conforma el identificador del caso de prueba.
- **Descripción:** Acción que debe realizar el sistema.
- **Condiciones de ejecución:** Describe las características y elementos que debe contener el sistema para realizar el caso de prueba.
- **Entrada:** Incluye las entradas necesarias para realizar el sistema.
- **Pasos de ejecución:** Pasos para realizar el caso de prueba**.**
- **Resultados Esperados:** Descripción de la respuesta del sistema ante el caso de prueba.
- **Resultado Obtenido:** Respuesta visual del sistema después de realizar el caso de prueba.
- **Evaluación de la prueba:** Clasificación de la prueba en satisfactoria o insatisfactoria.

A continuación se relaciona las pruebas de aceptación para la 1ra Iteración:

Tabla 18 Obtener la conexión entre AsiXMec y la clase GeomTool.

Pruebas de aceptación	
Código: HU1_P1	**HU: 1**
Nombre: Comprobar conexión entre AsiXMec y la clase GeomTool.	
Descripción:	
Condiciones de ejecución: La clase GeomTool recibe al menos dos entidades cónicas que están creadas.	

Pasos de ejecución: 1. Seleccionar una de las entidades cónicas. 2. Oprimir tecla “C”.
Resultados Esperados: Se muestra el mensaje “Se han recibido las entidades”.
Resultado Obtenido: Se muestra el mensaje “Se han recibido las entidades”.
Evaluación de la prueba: Satisfactoria

Tabla 19 Obtener la conexión entre GeomTool y la clase Intersect.

Pruebas de aceptación	
Código: HU2_P2	**HU: 2**
Nombre: Obtener la conexión entre GeomTool y la clase Intersect.	
Descripción:	
Condiciones de ejecución: La clase GeomTool recibe al menos dos entidades cónicas que están creadas.	
Pasos de ejecución: 1. Seleccionar una de las entidades cónicas. 2. Oprimir tecla “D”.	
Resultados Esperados: Se muestra el mensaje “Se han pueden hallar las intersecciones entre dos entidades”.	
Resultado Obtenido: Se muestra el mensaje “Se han pueden hallar las intersecciones entre dos entidades”.	
Evaluación de la prueba: Satisfactoria	

En los anexos se muestran las pruebas de aceptación de la segunda y tercera iteración. Estas iteraciones se relacionan a continuación en un gráfico (Figura 8).

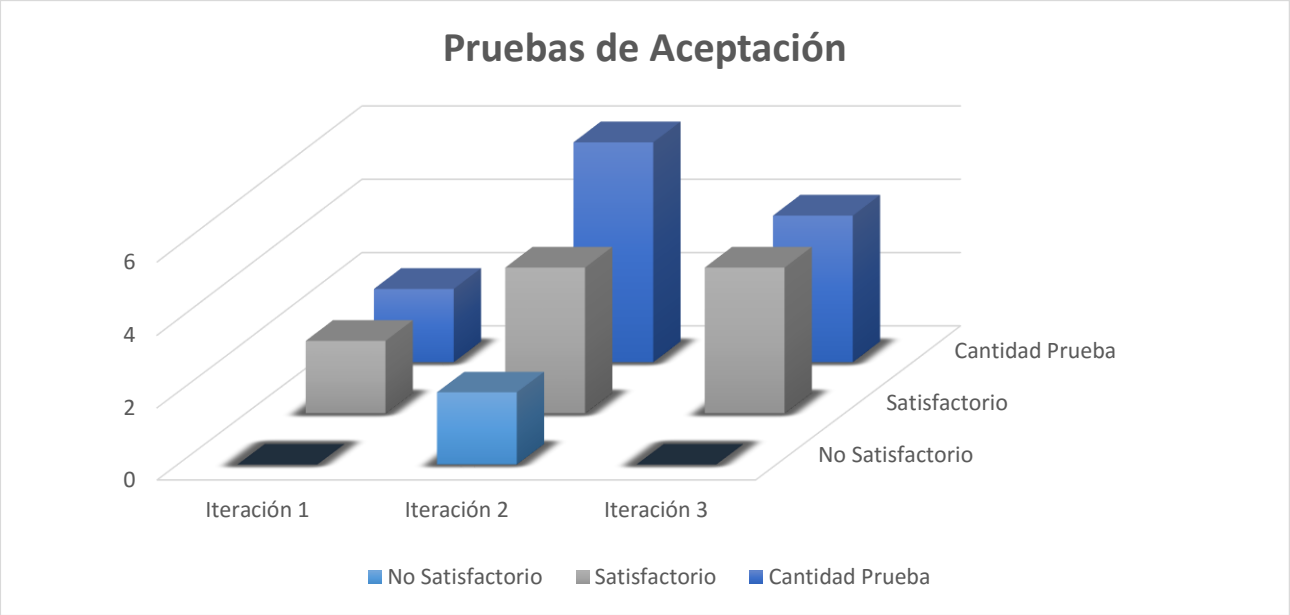

Figura 8 Pruebas de aceptación realizadas al componente de intersección. Fuente: Elaboración propia

Consideraciones del capítulo

Luego de desarrollar el capítulo se ha podido arribar a las siguientes conclusiones parciales:

- ✓ En la fase de desarrollo se definieron 8 tareas de ingeniería que organizaron la implementación del código.
- ✓ La realización de las pruebas de aceptación posibilitaron la obtención de un componente de intersección aceptado por el cliente.
- ✓ Del análisis del resultado obtenido se llegó a la conclusión de que se cumplió con el objetivo de la investigación al lograr correctamente el desarrollo del componente de intersección de entidades paramétricas.

Conclusiones generales

Durante el desarrollo de la investigación se planteó la necesidad de desarrollar un componente para obtener los puntos de intersección de entidades paramétricas en 2D para AsiXMec 2.0, dándole así cumplimiento al objetivo propuesto de la siguiente forma:

- ✓ La implementación de un método de intercepción de secciones cónicas basado en su representación matricial, y no en características específicas de cada tipo, permite incluir otras entidades no presentes en AsiXMec como las parábolas y las hipérbolas, solo es necesario obtener su representación matricial a partir de sus datos paramétricos.

Recomendaciones

Con el fin de contribuir a un mayor desarrollo de la presente investigación se realizan las siguientes recomendaciones:

- ✓ Realizar un estudio de las entidades spline para realizar la funcionalidad de intersectar dos entidades spline y las entidades spline-cónicas.

Referencias Bibliográficas

Markus Dieterle, Alternative-Computer-Programming.com . 2016. Alternative Computer Programming. [En línea] 2016.

Torres, J.C. *Diseño asistido por ordenador.* Universidad de Granada : s.n.

Albarrán Ligero, Justo. *FUNDAMENTOS DEL KBE (KNOWLEDGE BASED ENGINEERING). APLICACIÓN AL DISEÑO DE ENGRANAJES DE EJES PARALELOS CON CATIA V5. Capitulo 2.* Vol. Aplicación al diseño de engranajes de ejes paralelos con Catia v5.

Alpizar Naranjo, D.e.A.O y Iván. 2006. *El proceso Unificado Ágil v1.1.* 2006.

Arcia Salazar, Miguel. 2011. *Desarrollo de un componente visual para Qt utilizando el framework OpenCascade.* La Habana : s.n., 2011. TD_04510_11.

Benoît Jacob, Gaël Guennebaud. 2008. Eigen. [En línea] 2008. [Citado el: 16 de junio de 2017.] http://eigen.tuxfamily.org/index.php?title=Main_Page.

Bonilla, Ana. 2003. *GUIA TECNOPYME. Fase II. HERRAMIENTAS DE DISEÑO EINGENIERÍA.* Zamudio : ROBOTIKER, 2003.

Calvo, María del Carmen. *Parametrizaciones.*

Campuzano, Juan Carlos Ponce. 2016. GeoGebra. *The pencil of conics .* [En línea] International GeoGebra Institute , diciembre de 2016. https://www.geogebra.org/m/PB5wwqh3.

Conferencia: Enfoques de IS. Modelos y metodologías del PDS. **Paez, Andy Hernández. 2016.** UCI, La Habana : s.n., 2016.

CORDERO VALLE, JUAN M. 2002. *CURVAS Y SUPERFICIES PARA MODELADO GEOMÉTRICO.* s.l. : RA-MA EDITORIAL, 2002. 978-84-7897-531-0.

2007. elWebmaster. [En línea] Alejandra, 7 de noviembre de 2007. [Citado el: 8 de septiembre de 2016.] www.elwebmaster.com.

Estrada, José Ángel Lores. 2012. *MÓDULO DE TRANSFORMACIÓN A ENTIDADES 2D.* 2012.

González Olivera, Henry. 2016. *Entidad arco de elipse para el modelador geométrico de AsiXMec 2.0.* La Habana : s.n., 2016.

Hernández León, Rolando Alfred y Coello González, Sayda . 2011. *El proceso de investigación científica.* Editorial Universitaria : s.n., 2011.

Joskowicz, José. 2008. *Reglas y prácticas en Extreme Programing. Universidad.* España : s.n., 2008.

Lazo, Rojas. Junio de 2006. *Diseño asistido por computador.* s.l. : DISEÑO Y TECNOLOGÍA, Junio de 2006.

Lehmann, Charles H. *Geometría Analítica.* s.l. : Notiega Editors.

María José Pérez Pérez, Francisco J. González Cabrera. 2012. *Guía Comparativa de Metodologías Ágiles.* Universidad de Valladolid, E. U. de Informática (SEGOVIA). 2012.

McLean Hall, Gary. 2014. *Adaptive Code via C#: Agile coding with design patterns and SOLID principles.* Redmond, Washington : Microsoft Press, 2014. 978-0-7356-8320-4.

Moreno, María N. Gestión de recursos Informáticos del Departamento de Informatica y Automatica. [En línea] http://avellano.usal.es/~mmoreno/ASTema2.pdf.

Mørken, Knut. 2008. *Computing Intersections of Planar Spline Curves using Knot Insertion.* University of Oslo : s.n., 2008.

Nuñez, Dariel. 2016. *Módulo para la gestión de registros de entrada y salida en el Gestor de Documentos Administrativos Xabal eXcriba 3.1.* La Habana : s.n., 2016.

OPEN CASCADE S.A.S. 2015. Open CASCADE Technology 6.9.0. [En línea] mayo de 2015. http://www.opencascade.org.

Open CASCADE Technology 6.9.0. mayo 2015. *Modeling Algorithms.* mayo 2015.

—. mayo 2015. *Modeling Data.* mayo 2015.

Open CASCADE Technology. 2011. Open CASCADE Technology 6.5.0. [En línea] 03 de Marzo de 2011.

Palencia González, javier y García Llamas, María del Carmen. 2016. *Clasificación y Representación de Cónicas con Excel.* s.l. : Anales de ASEPUMA, 2016. ISSN-e 2171-892X.

Particle In Cell Consulting LLC, Woodland Hills, CA. 2013. Follow. [En línea] 10 de Agosto de 2013.

Pasadas Fernández, Miguel. 2017. *Resolución de Ecuaciones no Lineales.* Universidad de Granada : s.n., 2017.

Patricio Letelier, Mª Carmen Penadés. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Laboratorio de Sistemas de Información. Departamento de Sistemas Informáticos y Computación. Facultad de Informática. , Universidad Politécnica de Valencia. Valencia : s.n.

Prautzsch, Harmut. 2005. *Métodos de Bézier y B-Spline.* 2005. ISBN 3-937300-47-3.

Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera. *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

Rozental, Gennadiy. 2007. Boost Test Library. [En línea] Copyright ©, 2007. http://www.boost.org/doc/libs/1_36_0/libs/test/doc/html/index.html.

Sánchez, Rafael Álvarez. 2015. *Análisis y métodos numéricos con Geogebra.* 2015.

Sierra, Francisco Javier Ceballos. 2012. *Enciclopedia del lenguaje C++. s.l. .* 2012. 005.133 C387e RES..

Somerville, Ian. *Ingeniería de Software. Séptima edición.* Universidad de Alicante : s.n.

Sosa Arias, Julio César. Junio de 2014. *Base de datos de partes estandarizadas para un sistema CAD desarrollado sobre tecnologías.* La Habana : s.n., Junio de 2014.

Stack Exchange Inc. 2013. Mathematics Stack Exchange. [En línea] 2013. https://math.stackexchange.com/questions/316849/intersection-of-conics-using-matrix-representation.

—. 2013. Mathematics Stack Exchange. [En línea] 2013. https://math.stackexchange.com/questions/328300/decomposition-of-a-degenerate-conic.

Stewart, James. 2008. *Cálculo con Transendentes Tempanas. Parte 3.* La Habana : Felix Varela, 2008.

Targetware Informática S.A.C. software.com.ar. [En línea] http://www.software.com.ar/p/visual-paradigm-para-uml.

The QT Company Ltd. 2016. Sitio oficial de QT. [En línea] 2016. http://doc.qt.io/qt-5/linux.html.

Universidad de Coruña. 2015. *Tema 5: Capítulo 1. Cónicas.* Coruña : s.n., 2015.

WANG, SHYUE-WU. 2001. *An Efficient and Stable Ray Tracing Algorithm.* Institute of Information Science Academia Sinica Taipei : s.n., 2001.

Ziller, Eike. 2014. Sitio oficial QT. [En línea] febrero de 2014. http://blog.qt.io/blog/2014/02/05/qt-creator-3-0-1-released/.

Glosario de Términos

$curva\ cónica$[1]: “Se define sección cónica, también denominada cónica, como la curva resultante de intersecar un cono recto y un plano. Dependiendo del ángulo del plano que interseca respecto del eje de revolución del cono, que denominamos β , con $0 \leq \beta \leq \pi/2$ y del lugar por donde se produzca el corte se obtendrán las distintas secciones cónicas” (Palencia González, y otros, 2016)

$lapiz\ de\ cónicas$[2]: “Sean dos ecuaciones cónicas $f(x,y) = 0$ y g$(x,y) = 0$. Entonces el lápiz de las cónicas es el conjunto de curvas con ecuaciones $af(x,y) + bg(x,y) = 0$, donde a y b son números arbitrarios” (Campuzano, 2016).

Anexos 1 Pruebas Unitarias utilizando la biblioteca Boost

En las tablas de la 9 y 13 muestran las pruebas unitarias satisfactorias realizadas a los métodos de intersección línea-circunferencia, línea-elipse, circunferencia-circunferencia, circunferencia-elipse, elipse- elipse.

```
BOOST_AUTO_TEST_CASE(PruebaInterseccionLineaCircunferencia)
{
  std::cout << "PruebaInterseccionLineaCircunferencia test" << std::endl;

  Intersect* cint = new Intersect();

    Matrix3d E1, E2;
    E1 << 1,0,0,0,0,-0.5,0,0.5,0;
    E2 << 1,1,1,1,-2,0.5,1,0.5,1;

  std::vector<Eigen::Vector3d>  points = cint->intersectConics(E1,E2);

  BOOST_REQUIRE(points.size() == 2); //Verificar que exista los dos puntos de intersección

  for (auto p : points){
    double c1d = p.transpose() * E1 * p;
    double c2d = p.transpose() * E2 * p;
    BOOST_REQUIRE_SMALL(c1d, 1.0);//se garantiza que pertenezca a la 1era cónica
    BOOST_REQUIRE_SMALL(c2d, 1.0);//se garantiza que pertenezca a la 2da cónica

    }
}
```

Figura 9 BOOST_AUTO_TEST_CASE (PruebaInterseccionLineaCircunferencia). Fuente: Elaboración propia

```
BOOST_AUTO_TEST_CASE(PruebaInterseccionLineaElipse)
{
  std::cout << "PruebaInterseccionLineaElipse test" << std::endl;

  Intersect* cint = new Intersect();

    Matrix3d E1, E2;
    E1 << 1,0,0,0,0,-0.5,0,0.5,0;
    E2 << 19,-2,-8,-2,1,0,-8,0,4;

  std::vector<Eigen::Vector3d>  points = cint->intersectConics(E1,E2);

  BOOST_REQUIRE(points.size() == 2); //Verificar que exista los dos puntos de intersección

  for (auto p : points){
    double c1d = p.transpose() * E1 * p;
    double c2d = p.transpose() * E2 * p;
    BOOST_REQUIRE_SMALL(c1d, 1.0);//se garantiza que pertenezca a la 1era cónica
    BOOST_REQUIRE_SMALL(c2d, 1.0);//se garantiza que pertenezca a la 2da cónica
  }

}
```

Figura 10 BOOST_AUTO_TEST_CASE(PruebaInterseccionLineaElipse). Fuente: Elaboración propia

```
BOOST_AUTO_TEST_CASE(PruebaInterseccionCircunferenciaCircunferencia)
{
  std::cout << "PruebaInterseccionCircunferenciaCircunferencia test" << std::endl;

  Intersect* cint = new Intersect();

    Matrix3d E1, E2;
    E2 << 1,1,1,1,-2,0.5,1,0.5,1;
    E2 << 2,1,1,1,-3,-0.5,0,0.5,-2;

  std::vector<Eigen::Vector3d>  points = cint->intersectConics(E1,E2);

  BOOST_REQUIRE(points.size() == 2); //Verificar que exista los dos puntos de intersección

  for (auto p : points){
    double c1d = p.transpose() * E1 * p;
    double c2d = p.transpose() * E2 * p;
    BOOST_REQUIRE_SMALL(c1d, 1.0);//se garantiza que pertenezca a la 1era cónica
    BOOST_REQUIRE_SMALL(c2d, 1.0);//se garantiza que pertenezca a la 2da cónica

  }

}
```

Figura 11 BOOST_AUTO_TEST_CASE(PruebaInterseccionCircunferenciaCircunferencia).

Fuente: Elaboración propia

```
BOOST_AUTO_TEST_CASE(PruebaInterseccionCircunferenciaElipse)
{
  std::cout << "PruebaInterseccionCircunferenciaElipse test" << std::endl;

  Intersect* cint = new Intersect();

    Matrix3d E1, E2;
    E2 << 1,1,1,1,-2,0.5,1,0.5,1;
    E2 << 19,-2,-8,-2,1,0,-8,0,4;

  std::vector<Eigen::Vector3d>  points = cint->intersectConics(E1,E2);

  BOOST_REQUIRE(points.size() == 2); //Verificar que exista los dos puntos de intersección

  for (auto p : points){
    double c1d = p.transpose() * E1 * p;
    double c2d = p.transpose() * E2 * p;
    BOOST_REQUIRE_SMALL(c1d, 1.0);//se garantiza que pertenezca a la 1era cónica
    BOOST_REQUIRE_SMALL(c2d, 1.0);//se garantiza que pertenezca a la 2da cónica

  }

}
```

Figura 12 BOOST_AUTO_TEST_CASE(PruebaInterseccionCircunferenciaElipse). Fuente: Elaboración propia

```
BOOST_AUTO_TEST_CASE(PruebaInterseccionElipseElipse)
{
  std::cout << "PruebaInterseccionElipseElipse test" << std::endl;

  Intersect* cint = new Intersect();

    Matrix3d E1, E2;
    E1 << 1,0,0,0,0,-0.5,0,0.5,0;
    E2 << 19,-2,-8,-2,1,0,-8,0,4;

  std::vector<Eigen::Vector3d>  points = cint->intersectConics(E1,E2);

  BOOST_REQUIRE(points.size() == 2); //Verificar que exista los dos puntos de intersección

  for (auto p : points){
    double c1d = p.transpose() * E1 * p;
    double c2d = p.transpose() * E2 * p;
    BOOST_REQUIRE_SMALL(c1d, 1.0);//se garantiza que pertenezca a la 1era cónica
    BOOST_REQUIRE_SMALL(c2d, 1.0);//se garantiza que pertenezca a la 2da cónica

  }

}
```

Figura 13 BOOST_AUTO_TEST_CASE(PruebaInterseccionElipseElipse). Fuente: Elaboración propia

Anexos 2 Pruebas de aceptación de la segunda iteración

En las tablas de la 23 y 26 muestran las pruebas de aceptación satisfactorias realizadas.

Tabla 20 Intersectar dos entidades de tipo línea-línea

Pruebas de aceptación	
Código: HU2_P3	**HU: 2**
Nombre: Intersectar dos entidades de tipo línea-línea.	
Condiciones de ejecución: están creadas dos entidades cónicas de tipo línea que coincidan en un punto.	
Pasos de ejecución: 1. Seleccionar una de las curvas cónicas. 2. Oprimir tecla “P”.	
Resultados Esperados: Se muestra sobre las entidades seleccionada todos los puntos de intersección con la otra cónica.	
Resultado Obtenido: Se muestra sobre las entidades seleccionada los puntos de intersección con la otra cónica.	
Evaluación de la prueba: Satisfactoria	

Tabla 21 Intersectar dos entidades de tipo línea-circunferencia

Pruebas de aceptación	
Código: HU2_P4	**HU: 2**
Nombre: Intersectar dos entidades de tipo línea-circunferencia.	
Condiciones de ejecución: están creadas dos entidades cónicas de tipo línea y circunferencia que coincidan en al menos un punto.	

Pasos de ejecución: 1. Seleccionar una de las curvas cónicas. 2. Oprimir tecla “P”.
Resultados Esperados: Se muestra sobre las entidades seleccionada todos los puntos de intersección con la otra cónica.
Resultado Obtenido: Se muestra sobre las entidades seleccionada todos los puntos de intersección con la otra cónica.
Evaluación de la prueba: Satisfactoria

Tabla 22 Intersectar dos entidades de tipo línea-elipse

Pruebas de aceptación	
Código: HU2_P5	**HU: 2**
Nombre: Intersectar dos entidades de tipo línea-elipse.	
Condiciones de ejecución: están creadas dos entidades cónicas de tipo línea y elipse que coincidan en al menos un punto.	
Pasos de ejecución: 1. Seleccionar una de las curvas cónicas. 2. Oprimir tecla “P”.	
Resultados Esperados: Se muestra sobre las entidades seleccionada todos los puntos de intersección con la otra cónica.	
Resultado Obtenido: Se muestra sobre las entidades seleccionada todos los puntos de intersección con la otra cónica.	
Evaluación de la prueba: Satisfactoria	

Tabla 23 Intersectar dos entidades de tipo circunferencia-circunferencia

Pruebas de aceptación

Código: HU2_P6	**HU: 2**
Nombre: Intersectar dos entidades de tipo circunferencia-circunferencia.	
Condiciones de ejecución: están creadas dos entidades cónicas de tipo circunferencia que coincidan en al menos un punto.	
Pasos de ejecución: 1. Seleccionar una de las curvas cónicas. 2. Oprimir tecla “P”.	
Resultados Esperados: Se muestra sobre las entidades seleccionada todos los puntos de intersección con la otra cónica.	
Resultado Obtenido: Se muestra sobre las entidades seleccionada todos los puntos de intersección con la otra cónica.	
Evaluación de la prueba: Satisfactoria	

Anexos 3 Pruebas de aceptación de la tercera iteración

En las tablas de la 27 y 30 muestran las pruebas de aceptación satisfactorias realizadas.

Tabla 24 Intersectar dos entidades de tipo circunferencia-elipse

Pruebas de aceptación	
Código: HU2_P9	**HU: 2**
Nombre: Intersectar dos entidades de tipo circunferencia-elipse.	
Condiciones de ejecución: están creadas dos entidades cónicas de tipo circunferencia y elipse que coincidan en al menos un punto.	
Pasos de ejecución: 1. Seleccionar una de las curvas cónicas. 2. Oprimir tecla “P”.	
Resultados Esperados: Se muestra sobre las entidades seleccionada todos los puntos de intersección con la otra cónica.	
Resultado Obtenido: Se muestra sobre las entidades seleccionada todos los puntos de intersección con la otra cónica.	
Evaluación de la prueba: Satisfactoria	

Tabla 25 Intersectar dos entidades de tipo elipse-elipse

Pruebas de aceptación	
Código: HU2_P10	**HU: 2**
Nombre: Intersectar dos entidades de tipo elipse-elipse.	
Condiciones de ejecución: están creadas dos entidades cónicas de tipo elipse y elipse que coincidan en al menos un punto.	

Pasos de ejecución: 1. Seleccionar una de las curvas cónicas. 2. Oprimir tecla “P”.
Resultados Esperados: Se muestra sobre las entidades seleccionada todos los puntos de intersección con la otra cónica.
Resultado Obtenido: Se muestra sobre las entidades seleccionada todos los puntos de intersección con la otra cónica.
Evaluación de la prueba: Satisfactoria

Tabla 26 Intersectar dos entidades de tipo cónica

Pruebas de aceptación	
Código: HU1_P11	**HU: 1**
Nombre: Intersectar dos entidades de tipo cónica.	
Condiciones de ejecución: Están creadas dos entidades que coincidan en al menos un punto.	
Pasos de ejecución: 1. Seleccionar una de las entidades. 2. Oprimir tecla “P”.	
Resultados Esperados: Se muestra sobre la curva seleccionada todos los puntos de intersección con las otras entidades.	
Resultado Obtenido: Se muestra sobre la curva seleccionada todos los puntos de intersección con las otras entidades.	
Evaluación de la prueba: Se muestra sobre la curva seleccionada todos los puntos de intersección con las otras entidades.	
Evaluación de la prueba: Satisfactoria	

Tabla 27 Obtener los puntos de intersección entre una entidad y una lista de entidades

Pruebas de aceptación	
Código: HU1_P12	**HU: 1**
Nombre: Obtener los puntos de intersección entre una entidad y una lista de entidades.	
Descripción:	
Condiciones de ejecución: Están creadas dos entidades que coincidan en al menos un punto.	
Pasos de ejecución: 1. Seleccionar una de las entidades. 2. Oprimir tecla “P”.	
Resultados Esperados: Se muestra sobre la curva seleccionada todos los puntos de intersección con las otras entidades.	
Resultado Obtenido: Se muestra sobre la curva seleccionada todos los puntos de intersección con las otras entidades.	
Evaluación de la prueba: Se muestra sobre la curva seleccionada todos los puntos de intersección con las otras entidades.	
Evaluación de la prueba: Satisfactoria	