



**Universidad de las Ciencias Informáticas**

**Facultad 5**

*Trabajo de Diploma para optar por el Título de  
Ingeniero en Ciencias Informáticas.*

*Título*

*Módulo de creación paramétrica de cilindros y esferas 3D para la  
herramienta AsiXMeC 1.0.*

*Autor:*

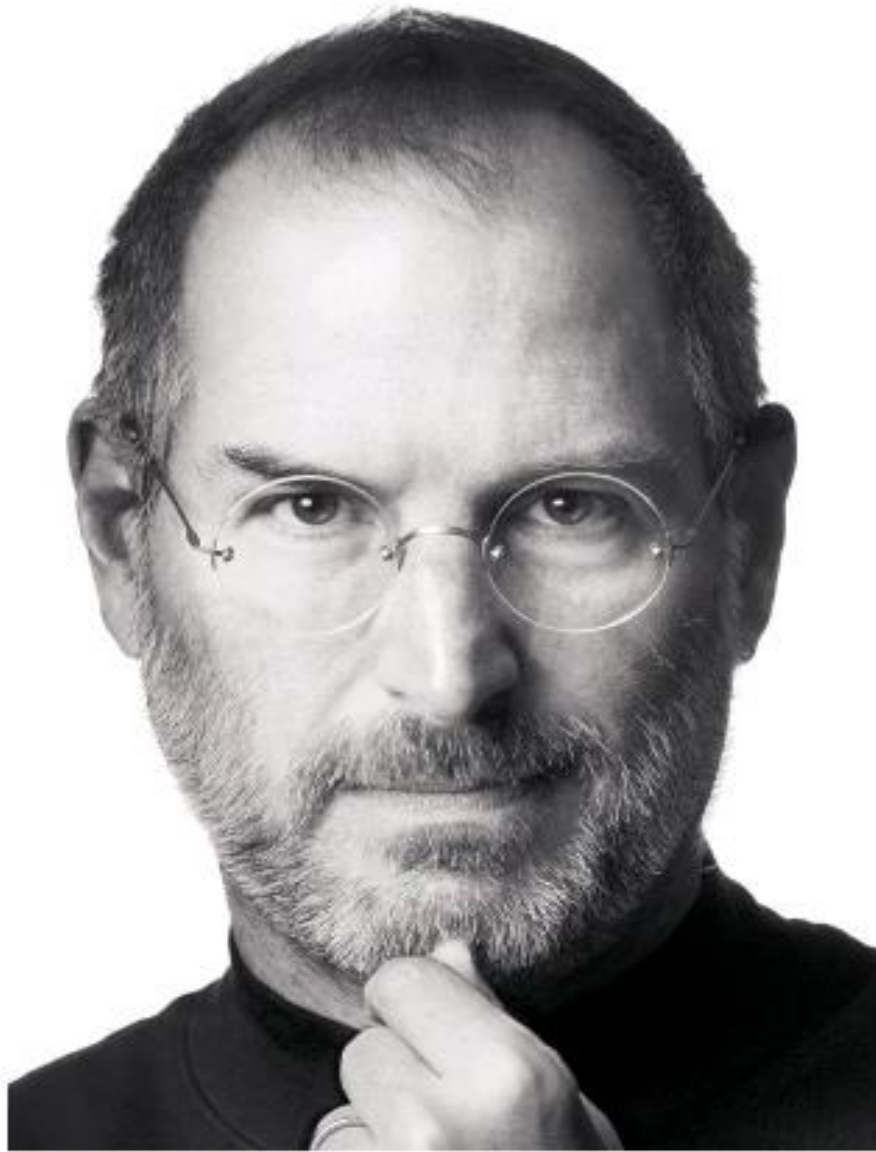
*Roberto Menejías García.*

*Tutores:*

*Ing. José Ángel Lores Estrada*

*Ing. Adrián Hernández Aguilera*

*Ciudad de La Habana, Julio 2016 "Año 58 de la Revolución"*



*“El hombre debe sentirse inútil cuando pasa un día sin aportar algo nuevo.”*

# *Agradecimiento*

---

Agradezco a mi familia, especialmente a mi madre Gilsy Rosa que ha sido siempre lo que más querido en toda mi vida.

Va dedicado a mi familia en general por el apoyo en todo momento. A mis amigos, profesores y conocidos, que de una manera u otra me han ayudado a lo largo de la carrera.

## **Declaración de auditoría.**

Por este medio declaramos que \_\_\_\_ es el único autor de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo. Para que así conste firmo la presente a los \_\_ días del mes de \_\_\_\_ del año \_\_\_\_.

---

Roberto Menejías García

Autor

---

Ing. José Ángel Lores Estrada

Tutor

## Resumen

El proyecto Diseño y Simulación de Estructuras Mecánicas del Centro de Entornos Interactivos 3D, se encuentra involucrado en la creación de la herramienta de diseño e ingeniería asistidos por computadora AsiXMeC 1.0, para brindar una solución libre y autónoma a la industria mecánica nacional. La herramienta tiene funciones de diseño para el modelado 3D que permiten crear algunos sólidos en el modelador geométrico 3D, utilizando operaciones de diseño sobre entidades 2D. Este proceso genera algunos sólidos básicos, pero el sistema no cuenta con una funcionalidad que genere las primitivas de forma directa en 3D, como lo hacen herramientas de este similares. Para solucionar esta necesidad se propone dotar a la herramienta de una funcionalidad que construya los sólidos primitivos básicos: cilindros y esferas específicamente, de forma automática y directa en el visor 3D.

Por lo que en el presente trabajo se propone integrar a la herramienta, un módulo para la creación de los sólidos primitivos básicos: cilindro y esfera en el visor 3D de AsiXMeC 1.0 de forma directa y paramétrica, usando la metodología de desarrollo AUP en su variación UCI para guiar el ciclo de vida de la solución.

Luego del análisis de la investigación se tomaron las mejores prácticas, conceptos, herramientas y tecnologías, que fueron utilizadas como punto de partida para el desarrollo de la solución propuesta, obteniendo como resultado un módulo de creación directa de los sólidos primitivos básicos antes mencionados, el cual fue incorporado a la herramienta AsiXMeC 1.0 cumpliendo con las expectativas y la arquitectura del sistema.

**Palabras claves:** sólidos, primitivas, modelador geométrico, modelado.

## Índice

<b>Introducción</b> .....	3
<b>Capítulo 1. Fundamentación Teórica</b> .....	8
1.1 Herramientas CAD.....	8
1.1.1 Herramientas homólogas. ....	9
1.2 Tecnología OpenCascade. ....	12
1.2.1 Framework OCAF.....	13
1.3 Modeladores geométricos .....	14
1.3.1 Modelador geométrico 2D( <i>sketcher</i> ) .....	14
1.3.2 Visor 3D. ....	14
1.4 Entorno de desarrollo Qt Creator.....	14
1.5 Lenguaje de programación C++.....	15
1.6 Metodología .....	16
1.7 Lenguaje Unificado de Modelado .....	17
1.8 Herramienta CASE .....	18
1.9 Sólidos 3D .....	18
1.9.1 Esfera .....	18
1.9.2 Cilindro.....	19
1.10 Conclusiones parciales. ....	20
<b>Capítulo 2. Descripción de la solución propuesta</b> .....	21
2.1 Propuesta de solución .....	21
2.1.1 Integración de la solución a la herramienta .....	21

2.2 Modelo conceptual.....	22
2.2.1 Descripción de las clases del modelo conceptual .....	23
2.3 Requisitos funcionales (RF) .....	24
2.4 Requisitos no funcionales (RNF).....	25
2.4.1 Descripción de requisitos no funcionales .....	25
2.5 Historias de usuarios .....	28
2.5.1 Descripción de las historias de usuarios.....	29
2.6 Arquitectura de la aplicación.....	31
2.7 Conclusiones parciales .....	33
<b>Capítulo 3. Características de la solución propuesta .....</b>	<b>34</b>
3.1 Módulo solid3D .....	34
3.2 Diagrama de clases del diseño .....	35
3.3 Descripción de las clases del diseño.....	37
3.4 Patrones de diseño .....	44
3.4.1 Patrones GRASP .....	44
3.4.2 Patrones GOF.....	45
3.5 Estándar de código .....	46
3.6 Algoritmos de la solución .....	47
3.7 Conclusiones parciales .....	49
<b>Capítulo 4: Pruebas del sistema .....</b>	<b>50</b>
4.1 Introducción .....	50
4.2 Pruebas.....	50
4.3 Casos de pruebas.....	50



4.4 Resultados de las pruebas .....	52
4.5 Conclusiones parciales .....	55
<b>Recomendaciones</b> .....	<b>57</b>
<b>Referencias Bibliográficas</b> .....	<b>58</b>
<b>Anexos</b> .....	<b>60</b>

Tabla 1. Clases del modelo conceptual .....	24
Tabla 2. Requisitos funcionales .....	25
Tabla 3. Descripción de RNF 1 .....	26
Tabla 4. Descripción de RNF 2 .....	27
Tabla 5. Descripción de RNF 3 .....	27
Tabla 6. HU1 Crear cilindro 3D .....	30
Tabla 7. HU2 Crear esfera 3D .....	31
Tabla 8. Descripción de las clases del cilindro.....	44
Tabla 9. Caso de pruebas del HU1 (tabla 1 de 2).....	52
Tabla 10. Caso de pruebas del HU1 (tabla 2 de 2) (Descripción de las variables del Caso de Prueba).....	52
Tabla 11. HU3 Editar cilindro 3D .....	60
Tabla 12. HU4 Editar esfera 3D .....	61
Tabla 13. HU5 Borrar cilindro 3D.....	61
Tabla 14. HU6 Borrar esfera 3D.....	62
Tabla 15. Descripción de las clases de diseño de la esfera .....	66
Tabla 16. Caso de prueba para la HU3 .....	66
Tabla 17. Caso de prueba para la HU3(Descripción de variables).....	66
Tabla 18. Caso de prueba para la HU2 .....	67
Tabla 19. Caso de prueba para la HU2(Descripción de variable) .....	67
Tabla 20. Caso de prueba para la HU4 .....	67
Tabla 21. Caso de prueba para la HU4(Descripción de la variable).....	68

# Índice de Ilustraciones

---

Ilustración 1. Iconos de primitivas.....	10
Ilustración 2. Cuadro de diálogo.....	11
Ilustración 3. Primitiva resultante del flujo .....	11
Ilustración 4. Modelo Conceptual. ....	23
Ilustración 5. Modelo de la arquitectura del módulo.....	32
Ilustración 6. Flujo del sistema .....	35
Ilustración 7 Diagrama de clases de cilindro .....	36
Ilustración 8. Código del método MakeDone (). .....	49
Ilustración 9. Crear cilindro .....	53
Ilustración 10. No conformidades por iteración.....	54
Ilustración 11. Nivel de aceptación. ....	55
Ilustración 12. Diagrama de clases de la esfera .....	63
Ilustración 13. Crear esfera 3D .....	68
Ilustración 14. Editar esfera 3D .....	69
Ilustración 15. Editar cilindro 3D .....	70

## Introducción

En la actualidad los procesos de la industria se desarrollan de forma automatizada o semi-automatizada, con el objetivo de optimizar tanto tiempo como recursos computacionales en los mismos. Esta automatización facilita además controlar de manera más eficiente y segura dichos procesos. El diseño industrial es una de las ramas más beneficiadas de la industria con la incorporación de las nuevas tecnologías, debido a un rápido incremento en el uso de potentes y eficaces herramientas para resolver muchos problemas de diseño, modelado de piezas y componentes industriales tanto en segunda como en tercera dimensión (2D y 3D respectivamente).

Estas herramientas de diseño asistido por computadora, por sus siglas en Inglés *Computer Aided Design* (CAD), son aplicaciones que hacen uso de programas computacionales para crear representaciones gráficas de objetos físicos ya sea en 2D o 3D, encargándose no solo de las tareas de diseño (como el dibujo técnico y la documentación del mismo) sino también se pueden realizar actividades de presentación del modelo y análisis del mismo (1). Los sistemas CAD agilizan los procesos de diseño, mejorando los prototipos digitales que se elaboran partiendo de un modelo para el diseño mecánico, convirtiéndose así, en un elemento esencial del flujo de trabajo para ingenieros, diseñadores y fabricantes. (2)

En Cuba también se trabaja en lograr la integración de las nuevas tecnologías a la industria, la automatización de los procesos industriales y el desarrollo de la informática dentro de las empresas. En la Universidad de la Ciencias Informáticas (UCI) se trabaja en aras de lograr la informatización de la industria nacional. Muestra de esto es el Centro VERTEX (Centro de entornos interactivos 3D) de la facultad 5, donde se desarrolló una primera versión de una herramienta CAD denominada AsiXMeC 1.0, que actualmente continúa en proceso de desarrollo.

Las herramientas CAD utilizan una interfaz denominada modelador geométrico 2D o 3D en caso de que se trabaje en el espacio. En los modeladores 2D son creados y visualizados los gráficos e imágenes que serán utilizados por los usuarios finales para elaborar sus diseños. En el diseño y modelado en tercera dimensión este modelador es el visor 3D, en el cual se visualizarán los objetos

que se creen en el espacio y se les realicen transformaciones para obtener un compuesto sólido lo más real posible (1). Los sólidos son objetos que se pueden crear en un visor, estos representan todo el volumen de un objeto que permite el análisis de propiedades físicas del objeto en sí. (3). Permiten realizar una representación real de un objeto, sin aproximaciones y obtener nuevos sólidos al aplicar operaciones de edición y diseño sobre otros sólidos. (4) Dentro de los objetos sólidos se encuentran las primitivas o sólidos primitivos básicos. Las primitivas son funciones que permiten la creación de sólidos con formas geométricas sencillas, como, por ejemplo, esferas y cilindros. (3)

Los sistemas CAD que trabajan el modelado 3D para elaborar compuestos sólidos más complejos, tienen varias formas definidas para generar las primitivas. La generación de primitivas define un conjunto de sólidos básicos tridimensionales, que pueden parametrizarse (disponen de elementos de control, modificables por el usuario). Son usadas frecuentemente por los diseñadores, en el diseño de objetos de alta complejidad, por ejemplo: tuercas, engranajes y tuberías. (4)

La herramienta AsiXMeC en su concepción sigue desde un principio, las pautas de diseño del sistema CAD Autodesk® Inventor®, donde en su visor 3D existe una sección para generar de forma directa primitivas. AsiXMeC 1.0, en su primera versión liberada, no cuenta con una funcionalidad que le permita crear los sólidos primitivos básicos en su visor 3D de manera directa. El procedimiento a seguir para la creación de sólidos en la herramienta, consta de un conjunto de pasos y restricciones, que requieren conocimientos de diseño. Estas restricciones obligan a tener que controlar manualmente las dimensiones de las entidades 2D que los generan, para garantizar dimensiones específicas en los sólidos obtenidos. Siguiendo este procedimiento, se pueden generar varias primitivas, pero no es posible construir algunos sólidos como la esfera y la pirámide. Por lo que se hace necesaria la implementación de una nueva funcionalidad para la creación de sólidos básicos: la esfera y el cilindro específicamente en AsiXMeC 1.0, obteniendo de forma más rápida y directa las primitivas predefinidas, y logrando definir un procedimiento similar al resto de los sistemas CAD, lo que haría a la herramienta más competitiva.

La situación problemática anteriormente mencionada permite plantear el siguiente **problema de la investigación**: ¿Cómo garantizar la creación de forma directa de los sólidos primitivos básicos Cilindro y Esfera en el visor 3D de la herramienta AsiXMeC 1.0?

Por lo que el **objeto de estudio de la investigación** se enmarca en la construcción paramétrica de los sólidos en el espacio 3D y el **campo de acción** se enfoca en la creación de los sólidos cilindros y esferas en un modelador geométrico utilizando la tecnología *OpenCascade*. Para darle solución al problema antes descrito se tiene como **objetivo general**:

Desarrollar un módulo para la creación de sólidos primitivos básicos: cilindros y esferas en la herramienta AsiXMeC 1.0, que sea paramétrico y que cumpla con sus pautas arquitectónicas.

Para lograr el objetivo propuesto se presentan como **tareas investigativas**:

1. Elaborar el estado del arte sobre las tendencias actuales de la creación de sólidos en las herramientas CAD, así como los principales conceptos que se emplean durante la investigación.
2. Seleccionar la metodología de desarrollo para definir los métodos y técnicas necesarias que guiarán el desarrollo del módulo como solución.
3. Describir las herramientas, tecnologías y lenguajes a emplear para definir el módulo en desarrollo.
4. Desarrollar la propuesta de solución, utilizando la arquitectura, los patrones y el diseño que se definieron para el desarrollo del módulo.
5. Realizar pruebas al módulo, validando la solución propuesta comparando los resultados obtenidos con los de la herramienta Autodesk® Inventor®.

Los métodos de investigación científica y procedimientos que se utilizaron para darle cumplimiento a las tareas anteriormente planteadas fueron:

## **Métodos teóricos empleados:**

**Método histórico:** para conocer la evolución y desarrollo de las herramientas CAD, identificando las características esenciales de su funcionamiento que pudieran ser de utilidad.

**Método lógico:** el método lógico utilizado fue el **método sistémico**, para lograr que todos los elementos de la concepción del modelo formen un todo que funcione de manera integrada a partir del estudio de las funcionalidades del elemento individual, y la adecuación de sus relaciones con los restantes (relaciones entre CAD – *OpenCascade* – OCAF – modeladores geométricos – sólidos 3D). (5)

## **Métodos empíricos.**

**Método de la observación científica y la medición** se emplean para la identificación del problema, el estudio objetivo del estado del arte, el diagnóstico y la evaluación de la aplicación.

**Método coloquial** se emplea para la presentación y discusión de los resultados en sesiones científicas (Seminarios de Tesis, Jornada Científica, Pre-defensa y Defensa). (5)

## **Procedimientos científicos.**

**Procedimiento de análisis** se emplea para la descomposición del objeto de estudio en partes y propiedades para el estudio de sus relaciones y componentes. Se complementa con la síntesis en la sistematización de los conocimientos mediante el entendimiento y descubrimiento de las relaciones esenciales y características generales entre las partes previamente analizadas. (5)

**Procedimiento de la abstracción** se emplea para la comprensión del objeto de estudio. (5)

Los **aportes prácticos esperados** con la elaboración del sistema son los siguientes:

- Integración del módulo de creación de los sólidos primitivos básicos 3D: cilindros y esferas a la herramienta AsiXMeC 1.0, cumpliendo con la arquitectura paramétrica de la misma.

- Permita a los mecánicos saltarse los pasos de diseño para la creación de sólidos primitivos 3D cilindro y esfera, haciendo omisión al proceso de esbozado 2D, medición y extrusión/revolución.

El presente trabajo está estructurado en 4 capítulos, un resumen, una introducción, conclusiones, recomendaciones, referencias bibliográficas, y los anexos. A continuación, se describen los principales aspectos abordados en cada uno de los capítulos:

## **Capítulo I: Fundamentación teórica.**

En este capítulo se puntualizan los principales conceptos relacionados con el tema, se realiza el estudio del estado del arte sobre el sistema CAD Autodesk® Inventor® y se elabora el marco teórico de la investigación.

## **Capítulo II: Descripción de la solución propuesta**

En este capítulo se realiza la modelación detallada y la construcción de la estructura de la solución. Se define la estructura y los elementos del diseño. Además, se hace un análisis del modelo del negocio correspondiente al sistema. Se realiza el levantamiento de los requisitos funcionales y no funcionales y se formalizan los artefactos derivados de la metodología de desarrollo de software que se seleccione.

## **Capítulo III: Características del sistema**

En este capítulo se abordan aspectos relacionados con la implementación del sistema en base a la arquitectura de desarrollo de software, su diagrama de clases del sistema y los patrones de diseño utilizados.

## **Capítulo IV: Pruebas al sistema**

Se documentan las pruebas de software al módulo de creación de los sólidos primitivos básicos 3D: cilindros y esferas de forma directa, para verificar que responda a un correcto funcionamiento y detectar posibles fallas en el sistema.



## Capítulo 1. Fundamentación Teórica.

En el capítulo se presentan los conceptos, definiciones y términos que ayudan a la comprensión y desarrollo de la solución propuesta. Se estudian las herramientas CAD con sus principales características, específicamente Autodesk® Inventor®, las tecnologías, metodologías y arquitecturas que brindan conocimiento y apoyo en el proceso de elaboración del producto final.

### 1.1 Herramientas CAD.

El diseño industrial desde sus inicios presentó problemas con la calidad y exactitud de sus modelos y bocetos, ya que estos eran realizados de forma manual o con herramientas rústicas, que propiciaban errores en los resultados finales. Una vez que el producto era terminado y vendido, más adelante aparecían problemas de diseño, dibujo y modelado que no podían ser resueltos.

Ante todos estos problemas se crearon varias herramientas que utilizan sofisticadas técnicas para lograr un diseño con menos errores, los sistemas CAD son ejemplo de estas herramientas. Esta tecnología, concebida para guiar el proceso de diseño gráfico y modelado del producto, también abarca tareas de manejo de bases de datos para el diseño y la fabricación.

El concepto de “diseño asistido por computadora” representa el conjunto de aplicaciones informáticas que permiten a un diseñador “definir” el producto a fabricar. Gracias a estas tecnologías se consigue una mayor productividad en el trazado de planos, integración con otras etapas del diseño, mayor flexibilidad, mayor facilidad de modificación del diseño, ayuda a la estandarización, disminución de revisiones y mayor control del proceso de diseño. (3)

En diseño industrial los sistemas CAD son utilizados principalmente para la creación de modelos de superficie o sólidos en 3D, o bien, dibujos de componentes físicos basados en vectores en 2D. Sin embargo, estos sistemas también se utilizan en los procesos de ingeniería desde el diseño conceptual y la unión de productos, a través de la aplicación de fuerzas y análisis dinámico de ensambles, hasta la definición de métodos de diseño. Esto le permite al ingeniero, desarrollador y/o diseñador analizar interactiva y automáticamente las variantes de diseño, para encontrar el diseño óptimo para crear mientras se minimiza el uso de prototipos físicos. (6)

## 1.1.1 Herramientas homólogas.

Existen muchas aplicaciones CAD en el mercado internacional, pero el equipo de desarrollo de AsiXMeC 1.0 ha seguido la filosofía de la herramienta Autodesk® Inventor®, como guía para el diseño e implementación. Por lo que el estudio del arte en busca de posibles soluciones existentes se centra específicamente en el flujo con el que se ejecuta la operación de crear primitivas de forma directa y paramétrica en este sistema CAD.

### Autodesk® Inventor®

La herramienta CAD Autodesk® Inventor® ofrece una gama completa y flexible de programas para diseño mecánico en 3D, simulación de productos, creación de herramientas y comunicación de diseños. Inventor combina un entorno de intuitivo de diseño 3D en *Windows*, para crear piezas y ensamblajes. Permite a los ingenieros y diseñadores enfocarse en la función de diseño para dirigir la creación automática de componentes avanzados, como estructuras mecánicas, tuberías, tubos, estructura rotativa o cableado eléctrico. (7)

La última versión (año 2016) de la herramienta crea los sólidos primitivos básicos de distintas formas. Una primera variante que incluye al realizar operaciones de diseño sobre entidades 2D, dicho boceto 2D está vinculado a la operación resultante (sólido 3D), de modo que, si se edita el boceto, la operación se actualiza; otra vía es modelar sólidos importados de otros sistemas CAD en formato SAP, pero estos modelos no son igual a los sólidos de Inventor, los cuales pueden tener acceso a sus bocetos 2D, operaciones, cotas y restricciones que se utilizaron para crear el sólido base; y por último la vía que permite crear los sólidos básicos de forma directa saltándose los pasos de generar un boceto 2D.

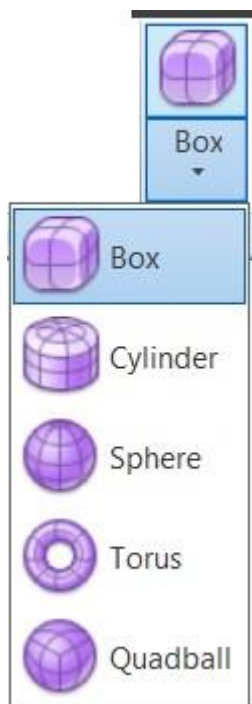
### Selección de la funcionalidad para implementar la solución propuesta

Teniendo en cuenta el objetivo del trabajo, se definirá para AsiXMeC 1.0 la funcionalidad con las características que permiten la construcción de los sólidos primitivos básicos de forma directa en 3D sin que se generen bocetos 2D y se apliquen operaciones de diseño. Se seguirá el flujo que

# Capítulo 1: Fundamentación Teórica

tiene esta funcionalidad en la herramienta Autodesk® Inventor®, ya que es el sistema CAD por el que se ha guiado el desarrollo de AsiXMeC 1.0 para implementar muchas de sus funcionalidades. Las siguientes imágenes describen de forma breve el flujo que sigue el sistema Autodesk® Inventor®.

Primeramente, se selecciona el icono de la primitiva que se desea crear en la barra de modelado 3D de la herramienta.



**Ilustración 1. Iconos de primitivas**

Al escoger el sólido básico que se va a construir, se levanta un diálogo para el dimensionado de las variables que definen los elementos del sólido. Pero antes de fijar los valores, se debe seleccionar el plano donde se desea crear la primitiva.

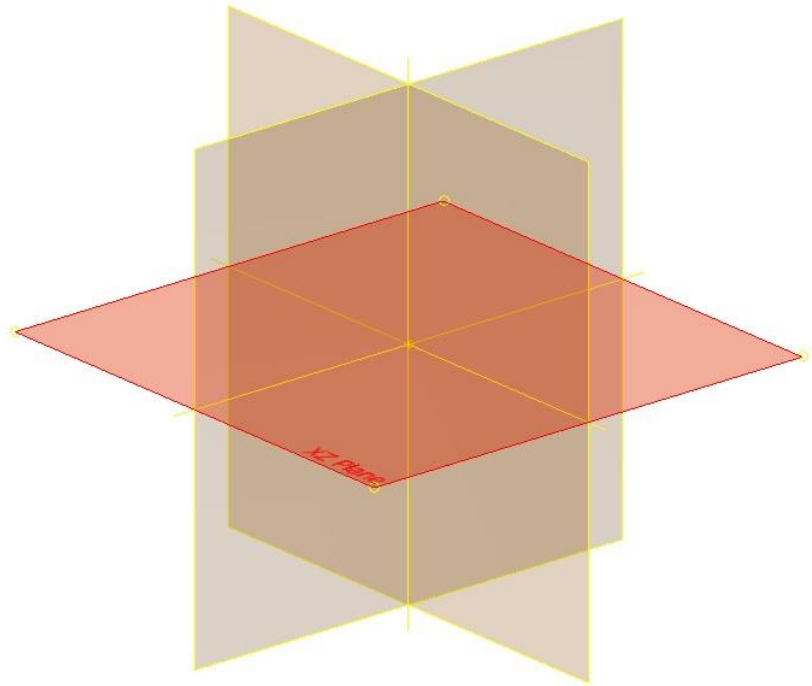
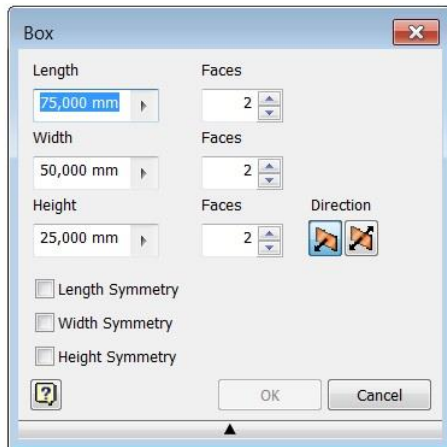


Ilustración 2. Cuadro de diálogo

Una vez seleccionado el plano y fijados los valores de las variables, se construye el sólido primitivo básico en el área de trabajo pulsando el botón *OK* del diálogo.

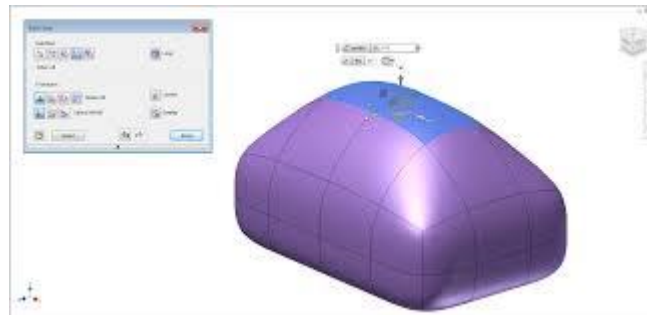


Ilustración 3. Primitiva resultante del flujo

# Capítulo 1: Fundamentación Teórica

---

La primitiva creada en el área de trabajo por esta funcionalidad es un sólido primitivo básico (en forma libre). A este sólido se le pueden aplicar las operaciones de diseño y edición que tiene la herramienta.

Este procedimiento no genera ningún boceto 2D asociado al sólido obtenido por lo que el flujo que seguirá la propuesta de solución, es similar a la funcionalidad descrita anteriormente. Los siguientes pasos fueron los definidos para la creación de forma directas de los sólidos primitivos básicos en la herramienta AsiXMeC 1.0, basados en el resultado de la investigación:

1. Pulsar el icono del sólido en el *ribbon* (barra de modelado 3D con las operaciones).
2. Seleccionar el punto central del sólido.
3. Introducir los valores de las variables de los elementos del sólido.
4. Pulsar botón *Done* para construir el sólido.

Para la implementación de esta funcionalidad en AsiXMeC 1.0 se utilizaron las siguientes tecnologías y herramientas.

## 1.2 Tecnología OpenCascade.

La biblioteca *OpenCascade* (abreviatura de las siglas en inglés *Computer Aided Software for Computer Aided Design and Engineering*), es un software de código abierto para el desarrollo de plataformas 3D y especializada en creación de aplicaciones CAD, para el diseño y simulación de objetos y piezas enfocado a ingenieros y profesionales. La tecnología permite crear superficies sólidas en 3D, desarrollar aplicaciones y simular ensayos (8). Es la tecnología con la cual se ha desarrollado la herramienta hasta la actualidad.

La tecnología *OpenCascade* utiliza un *framework* que facilita su uso y vuelve el trabajo con la biblioteca más rápido, el cual ha sido utilizado en el desarrollo de la herramienta desde sus inicios. A continuación, se dará una breve caracterización del mismo.

## 1.2.1 Framework OCAF

OCAF por sus siglas en inglés significa “*Open CASCADE Application Framework*”. Es una plataforma fácil de usar que permite el desarrollo rápido de aplicaciones sofisticadas de diseño. Una aplicación típica desarrollada con OCAF puede tener un modelador geométrico en 2D o 3D, herramientas de fabricación o de análisis, y aplicaciones de simulación o herramientas de ilustración (9). Cuenta con un sinnúmero de características que la convierten en una de las opciones más atractivas a la hora de trabajar en el desarrollo de herramientas CAD, características como:

- Facilidad de diseño de la arquitectura de la aplicación.
- Definición de los componentes y la forma en que cooperan.
- Definición del modelo de datos capaz de soportar las funcionalidades requeridas.
- Sincronizar la visualización con los datos. Los comandos de visualización de objetos tienen que actualizar la vista una vez que estos son llamados.
- Soportar comandos para las operaciones “deshacer” y “rehacer” (*Ctrl+Z*). Esta función debe ser tomada en cuenta muy tempranamente en el proceso de diseño, y hecho así, funciona de manera eficiente.
- Implementación de las funciones para guardar los datos. Si la aplicación tiene un ciclo de vida muy largo, la compatibilidad de los datos entre las versiones tiene que ser tratada.
- Facilidad para crear una interfaz de usuario.

*OpenCascade* y OCAF utilizan como interfaz visual para el manejo de las entidades, una herramienta conocida como modelador geométrico, que por sus características hace que el resultado final resulte muy sencillo e intuitivo para el usuario.

## 1.3 Modeladores geométricos

A continuación, se muestra una breve descripción de los principales modeladores geométricos utilizados en las herramientas CAD y por ende en el sistema AsiXMeC 1.0. Los modeladores brindan interfaces para trabajar en el entorno 2D y 3D.

### 1.3.1 Modelador geométrico 2D(*sketcher*)

Los modeladores geométricos 2D se basan en entidades geométricas vectoriales como puntos, líneas, arcos y polígonos, con las que se puede operar a través de una interfaz gráfica. Son una herramienta muy poderosa que permiten modelar de una manera sencilla todo tipo de objetos. Estimulan la creatividad y la capacidad intelectual del usuario respecto al manejo de la geometría, e introducen y refuerzan el conocimiento de las tres dimensiones; de ahí la decisión de utilizarlos como interfaz visual para las herramientas CAD, y por supuesto para la herramienta AsiXMeC 1.0. (6)

### 1.3.2 Visor 3D.

Es una interfaz que sirve como área de trabajo del usuario para el entorno tridimensional sobre el cual han de poderse aplicar operaciones y transformaciones a los sólidos creados de forma directa o a partir de entidades 2D. En este se visualiza el cuerpo final de la pieza o producto que se diseñó, por lo que su desarrollo y funcionalidad es primordial para toda herramienta CAD, y por lo tanto también para AsiXMeC 1.0. El visor 3D es una forma de representar objetos del mundo real en un modo más natural, al representar ancho, altura y profundidad. Este modelador geométrico 3D utiliza los ejes X, Y, Z. (6). A continuación, se describe el entorno de desarrollo que se utilizará para la implementación de la funcionalidad.

## 1.4 Entorno de desarrollo Qt Creator

Se trata de un *framework* para el desarrollo de aplicaciones que requieran una interfaz gráfica de usuario. El IDE (Entorno de Desarrollo Integrado) proporciona una excelente compatibilidad para

desarrollar varias aplicaciones en, Microsoft Windows: 98, 2000, XP, NET 4.0 y en Linux entre otras. (10)

Las librerías de *Qt Creator* están disponibles no solo para C++, sino también ofrece soluciones para utilizarse con otros lenguajes tales como:

- Java (QJambi)
- Python (PyQt)
- Ruby
- JavaScript (módulo QtScript)
- PHP

Para desarrollar aplicaciones en esta plataforma, bastará con un simple editor de texto, sin embargo, se pueden utilizar herramientas o editores de texto con características más avanzadas y robustas. Actualmente existe un *plugin* para desarrollar aplicaciones Qt utilizando el entorno Eclipse, el cual es conocido por contar con *plugins* para desarrollar en PHP, C++, trabajar con SQL y otras. Qt como toda buena herramienta, puede autocompletar, marcar sintaxis y está muy bien integrado con *Qt Designer* y la documentación Qt (10). El entorno de desarrollo se utilizará junto con C++ como lenguaje de programación, del cual se brinda una breve descripción a continuación.

## 1.5 Lenguaje de programación C++

Está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo, es a su vez uno de los que menos automatismos traen, con lo que obliga a usar bibliotecas de terceros. De los lenguajes de alto nivel, se puede decir que es uno de los más cercanos al lenguaje de máquina, lo que le proporciona mayor velocidad de ejecución con respecto a otros. C++ es un lenguaje de programación de propósito general. Todo



puede programarse con este lenguaje, desde sistemas operativos y compiladores hasta aplicaciones de bases de datos, procesadores de texto o juegos (11).

## 1.6 Metodología

Para guiar el proceso de desarrollo de la solución propuesta se escogió la Metodología Unificada Ágil (AUP), haciendo una variación de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.

AUP es una forma simplificada del Proceso Racional Unificado (*RUP, Rational Unified Process*) desarrollada por *Scott Ambler*. Describe un enfoque simple del desarrollo del software usando técnicas y conceptos ágiles. Algunas técnicas usadas por AUP incluyen el desarrollo orientado a pruebas, modelado y gestión de cambios ágiles y refactorización de base de datos para mejorar la productividad (12).

AUP establece cuatro fases: (Inicio, Elaboración, Construcción, Transición). La UCI decide para el ciclo de vida de los proyectos mantener la fase de Inicio, en la cual se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto. Se unifican las restantes 3 fases de AUP en una sola, a la que se denomina fase de Ejecución que sería aquella fase en la que se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software, y se agrega una fase de cierre. En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (12).

Algunas de las ventajas que proporciona la metodología AUP son (13):

- El personal sabe lo que está haciendo: no obliga a conocer detalles.
- Simplicidad: apuntes concisos.
- Agilidad: procesos simplificados del RUP.
- Centrarse en actividades de alto valor: esenciales para el desarrollo.
- Herramientas independientes: a disposición del usuario.
- Fácil adaptación de este producto.

A continuación se muestran algunas desventajas que presenta dicha metodología (13):

- Como es un proceso simplificado, muchos desarrolladores eligen trabajar con RUP, por tener a disposición más detalles en el proceso.

## 1.7 Lenguaje Unificado de Modelado

Los lenguajes de modelado visual permiten especificar, construir, documentar y visualizar artefactos de un sistema de software. Se seleccionó para el presente trabajo la utilización del lenguaje UML. El mismo está compuesto por diversos elementos gráficos que se combinan para conformar diagramas.

La selección del lenguaje vino dada por ser un lenguaje hasta cierto punto universal, pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. La especificación de UML no define un proceso estándar, pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos (14).

## 1.8 Herramienta CASE

*Visual Paradigm* es una herramienta para UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. (14)

*Visual Paradigm for UML* es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación y pruebas. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (*Visual Paradigm*) Teniendo en cuenta sus características y los beneficios que brinda para la construcción de software, especialmente referente al modelado, se decidió utilizar *Visual Paradigm for UML* para el modelado. Además de ello, se tuvo en cuenta que esta constituye la herramienta que utiliza la Universidad. (14)

## 1.9 Sólidos 3D

Los sólidos primitivos básicos 3D en las principales herramientas de diseño asistido por computadoras del mercado, varían en su forma y número. Entre los más comunes o los que más se encuentran definidos en estas aplicaciones están las esferas y los cilindros, primitivas de las cuales se dará una breve descripción en este acápite para una mejor comprensión de la solución que se propone.

### 1.9.1 Esfera

Una superficie esférica es una superficie de revolución, viene a ser el conjunto de los puntos del espacio cuyos puntos equidistan de otro punto fijo llamado centro. Los puntos cuya distancia es menor que la longitud del radio, forman el interior, y cada punto se llama punto interior de la superficie esférica. La unión del interior y la superficie esférica se llama bola cerrada, o esfera, en la geometría elemental del espacio. Obviamente, la esfera es un sólido geométrico (15).

A continuación, se muestran algunas de las principales propiedades con las que cuenta el sólido:

- Cualquier segmento que contiene el centro de la esfera y sus extremos en la superficie esférica, es un diámetro.
- Cualquier sección plana de una esfera es un círculo.
- Cualquier sección que pasa por el centro de una esfera es un círculo mayor, y si la sección no pasa por el centro es un círculo menor.
- Si se da un círculo de una esfera, los extremos del diámetro perpendicular a aquel se llaman polos.

## 1.9.2 Cilindro

Es una superficie de las denominadas cuadráticas formada por el desplazamiento paralelo de una recta llamada generatriz a lo largo de una curva plana, denominada directriz del cilindro.

Si la directriz es un círculo y la generatriz es perpendicular a él, entonces la superficie obtenida, llamada cilindro circular recto, será de revolución y tendrá por lo tanto todos sus puntos situados a una distancia fija de una línea recta, el eje del cilindro. El sólido encerrado por esta superficie y por dos planos perpendiculares al eje también es llamado cilindro. Este sólido es utilizado como una superficie Gaussiana (15).

Un cilindro puede ser:

- Rectangular: si el eje del cilindro es perpendicular a las bases.
- Oblicuo: si el eje no es perpendicular a las bases.
- De revolución: si está limitado por una superficie que gira  $360^\circ$  grados.

El cilindro que implementará la solución propuesta de los descritos anteriormente, será el cilindro rectangular.

## 1.10 Conclusiones parciales

- El uso de la tecnología *OpenCascade* y el *framework* OCAF es una buena elección para realizar un software CAD por sus características y prestaciones. Han sido utilizados para el desarrollo de AsiXMeC 1.0 desde sus inicios y es la tecnología en que está montado el sistema.
- La funcionalidad de creación directa de sólidos primitivos básicos 3D es ampliamente utilizada y necesaria en los softwares CAD utilizados en la actualidad, y su implementación reviste gran importancia en la utilidad final de los mismos.

## **Capítulo 2. Descripción de la solución propuesta**

En este capítulo se abordará parte de lo referente a la solución propuesta, describiendo el ciclo seguido como parte de la ingeniería de software desde el modelo de dominio, pasando por los requisitos funcionales y no funcionales, hasta la descripción de las historias de usuarios derivados de dichos requisitos funcionales; tomándose como base la metodología AUP variación UCI como guía para el proceso de desarrollo. También se definirá la arquitectura que se usará para la implementación del módulo como propuesta de solución.

### **2.1 Propuesta de solución**

AsiXMeC 1.0 no cuenta con una funcionalidad que permita la creación de los sólidos primitivos básicos 3D cilindros y esferas de forma directa en su visor, obligando al diseñador a construir las primitivas utilizando solamente el procedimiento existente para crear los cilindros, y siendo imposible la creación de esferas.

Para resolver este problema se propone desarrollar un módulo que permita crear de forma directa las primitivas en el visor de la herramienta, para brindar a los diseñadores una forma sencilla, directa y precisa de obtener dichos sólidos en el proceso de diseño. Permitiendo además que los mismos puedan ser construidos con las dimensiones deseadas por el usuario, ya que podrá introducir los valores de sus elementos, en forma de parámetros. A partir de las características de la solución propuesta desarrollada, el usuario podrá modificar los sólidos creados y eliminarlos cuando desee.

#### **2.1.1 Integración de la solución a la herramienta**

El proceso existente en AsiXMeC 1.0 para la construcción de los sólidos 3D, está basado en un flujo de trabajo que incluye varios módulos de la herramienta, mediante su interacción, logran construir un modelo en tercera dimensión, realizando varias operaciones para ello.

La nueva funcionalidad permitirá crear las primitivas básicas 3D de una manera paramétrica y directa por AsiXMeC 1.0, y para lograrlo estará en interacción con los siguientes módulos de la herramienta:

## *Capítulo 2: Descripción de la Solución Propuesta*

---

- **Asixmec:** módulo principal de la herramienta, cuenta con la interfaz principal y todos los entornos de trabajo, en los cuales los diseñadores trabajan e interactúan con la aplicación. Este módulo, visualiza el producto final elaborado en el proceso de diseño.
- **Cad-core:** módulo que contiene el árbol de OCAF. Dicho árbol almacena todas las entidades y operaciones que se construyan en la herramienta en tiempo de ejecución en forma de nodos. Cada nodo contiene la información necesaria para crear, modificar y eliminar, así como realizar operaciones de diseño sobre cada objeto.

A continuación, se describen los conceptos asociados a la realidad que trata la propuesta de solución anteriormente definida.

### **2.2 Modelo conceptual**

Es un conjunto de conceptos y de reglas destinados a representar de forma global los aspectos lógicos de los diferentes tipos de elementos existentes en la realidad que está siendo analizada. Es una representación figurada de una experiencia empírica, que tiene como objetivo ayudar a comprender la realidad. (16)

## Capítulo 2: Descripción de la Solución Propuesta

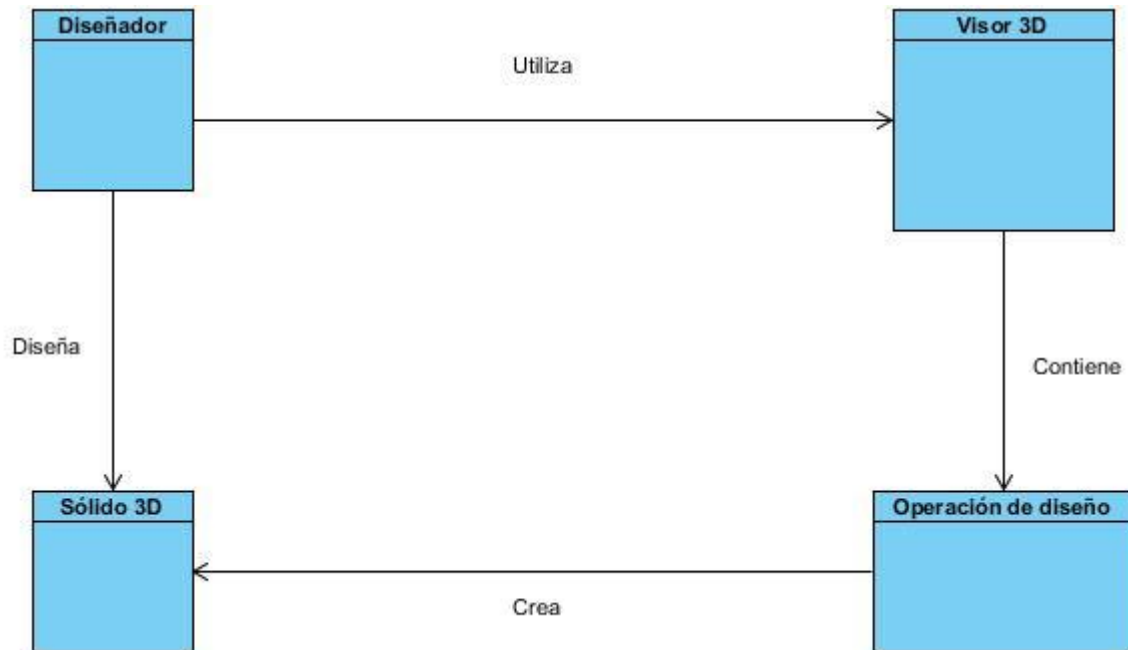


Ilustración 4. Modelo Conceptual.

### 2.2.1 Descripción de las clases del modelo conceptual

Nombre	Descripción
<b>Diseñador</b>	Usuario con conocimiento de diseño, que utiliza la herramienta para elaborar modelos y bocetos de figuras en tercera dimensión. El diseñador crea las primitivas y las utiliza en el visor.
<b>Visor 3D</b>	Modelador geométrico para el trabajo en tercera dimensión. Epígrafe 1.7.2
<b>Operación de diseño</b>	Contiene todas las operaciones para construir, editar y eliminar cada uno de los sólidos primitivos básicos en el visor 3D.



## Capítulo 2: Descripción de la Solución Propuesta

<b>Sólido 3D.</b>	Son todos los sólidos que pueden ser creados por el diseñador, a partir de las operaciones de diseño, en este caso son las esferas y los cilindros.
-------------------	---

Tabla 1. Clases del modelo conceptual

Una vez definida la propuesta de solución y sus conceptos asociados, se realiza el levantamiento de los requisitos, para definir las funcionalidades y características del sistema como solución.

### 2.3 Requisitos funcionales (RF)

Son declaraciones formales de las prestaciones que proporciona el sistema. Estas deben argumentar la manera en que la aplicación puede reaccionar a las entradas y cómo se debe comportar en situaciones particulares (17). El proceso de levantamiento de requisitos concluyó con un total de 6 requisitos funcionales, los cuales se enumeran a continuación y se describen más adelante con historias de usuario en el epígrafe 2.5:

No	Nombre	Descripción	Complejidad
RF1	Crear cilindro 3D	Se construye un cilindro en el visor 3D con los valores del radio y la altura introducidos por parámetros. Se genera el cilindro con esas dimensiones una vez pulsado el botón <i>Done</i> en el diálogo.	Alta
RF2	Crear esfera 3D	Crea una esfera con su radio de acuerdo con el valor que fue introducido por el diseñador como parámetro. Esto se realiza una vez levantado el diálogo.	Alta
RF3	Editar cilindro 3D	Permite editar el cilindro una vez esté creado en el visor 3D. Modifica los valores de sus elementos, con los cuales fue creado en un inicio.	Alta
RF4	Editar esfera 3D	Después de construida la esfera en el visor 3D, edita su dimensión,	Alta

## Capítulo 2: Descripción de la Solución Propuesta

		modificando el radio con el que fue construido al inicio.	
RF5	Borrar cilindro 3D	Permite eliminar un cilindro del visor 3D de la herramienta y su representación en el navegador de objetos.	Alta
RF6	Borrar esfera 3D	Permite eliminar una esfera del visor 3D de la herramienta y su representación en el navegador de objetos.	Alta

Tabla 2.Requisitos funcionales

### 2.4 Requisitos no funcionales (RNF)

Representan restricciones a los servicios que debe brindar el software (17). A continuación, se establecen los requisitos no funcionales establecidos para el desarrollo de la solución propuesta.

#### RNF 1. Usabilidad

##### RNF 1.1. Operabilidad

#### RNF 2. Portabilidad

##### RNF 2.1 Adaptabilidad

#### RNF 3. Eficiencia de rendimiento

##### RNF 3.1 Utilización de recursos

### 2.4.1 Descripción de requisitos no funcionales

<b>Atributo de Calidad</b>	Usabilidad.
<b>Sub-atributos/Sub-característica</b>	Operabilidad.
<b>Objetivo</b>	Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
<b>Origen</b>	Proveedor de requisitos.
<b>Artefacto</b>	Los diálogos pertenecientes a los requisitos

## Capítulo 2: Descripción de la Solución Propuesta

	funcionales del módulo.
<b>Entorno</b>	La herramienta AsiXMeC 1.0 desplegada.
<b>Estímulo</b>	<b>Respuesta: Flujo de eventos (Escenarios)</b>
<b>1. a. Número de pasos a ejecutar para llegar a los requisitos del sistema</b>	
Interactuar con los requisitos de la herramienta.	<ol style="list-style-type: none"> <li>1. La herramienta muestra la pantalla principal para iniciar, elaborar diseños o salir.</li> <li>2. El usuario selecciona la opción deseada.</li> <li>3. La herramienta verifica la opción seleccionada y muestra las pantallas siguientes de los escenarios que conforman la vía escogida.</li> <li>4. El usuario accede a iniciar, configurar opciones o salir con un solo clic.</li> </ol>
<b>Medida de respuesta</b>	
Navegar en el sistema.	

Tabla 3. Descripción de RNF 1

<b>Atributo de Calidad</b>	Portabilidad.
<b>Sub-atributos/Sub-característica</b>	Adaptabilidad.
<b>Objetivo</b>	Capacidad del módulo que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
<b>Origen</b>	Proveedor de requisitos.
<b>Artefacto</b>	El módulo.
<b>Entorno</b>	El módulo integrado.
<b>Estímulo</b>	<b>Respuesta: Flujo de eventos (Escenarios)</b>

## Capítulo 2: Descripción de la Solución Propuesta

<b>1. a. Capacidad del sistema de adaptarse de forma efectiva a diferentes entornos</b>	
El módulo está diseñado con tecnologías que permiten que este se adapte a las distribuciones de Linux, desde Ubuntu 12.04 en adelante, con arquitectura de x64 bits. Las mismas necesarias para que funcione la herramienta como tal.	N/A
<b>Medida de respuesta</b>	
El ambiente de despliegue de la herramienta.	

Tabla 4. Descripción de RNF 2

<b>Atributo de Calidad</b>	Eficiencia en el rendimiento
<b>Sub-atributos/Sub-característica</b>	Utilización de recursos
<b>Objetivo</b>	Grado en el que las cantidades y tipos de recursos utilizados por un producto o sistema, al realizar sus funciones cumplen con los requisitos.
<b>Origen</b>	Arquitecto de software.
<b>Artefacto</b>	El módulo de creación de cilindros y esferas 3D.
<b>Entorno</b>	El módulo integrado a la herramienta AsiXMeC 1.0
<b>Estímulo</b>	<b>Respuesta: Flujo de eventos (Escenarios)</b>
<b>1. a. Características de Hardware</b>	
<ul style="list-style-type: none"> <li>Las estaciones de trabajo donde se desplegará la herramienta tiene que tener al menos 2 GB de memoria RAM.</li> <li>Para la iteración con el módulo el computador debe tener los periféricos <i>mouse</i>(ratón) y teclado.</li> </ul>	El módulo de creación de cilindros y esferas 3D funcionando correctamente.

## Capítulo 2: Descripción de la Solución Propuesta

<b>Estímulo</b>	
<b>1.b. Características de Software</b>	
<ul style="list-style-type: none"><li>• Sistema operativo Linux, en sus distribuciones de x64 bits desde la versión Ubuntu 12.04 en adelante.</li></ul>	
<b>Estímulo</b>	
<b>1.c. Características de diseño e implementación</b>	
<ul style="list-style-type: none"><li>• IDE Qt <i>Creator</i> v5.0 como entorno de desarrollo.</li><li>• <i>Visual Paradigm for UML</i> v8.0</li><li>• Tecnología <i>OpenCascade</i> con su <i>framework</i> OCAF.</li><li>• AUP variación UCI como metodología de desarrollo.</li></ul>	
<b>Medida de respuesta</b>	
Desarrollo y funcionamiento del módulo de creación paramétrica de cilindros y esferas 3D.	

Tabla 5. Descripción de RNF 3

### 2.5 Historias de usuarios

A continuación, la descripción de las historias de usuarios por cada funcionalidad que se definió anteriormente.

Las historias de usuarios (HU) describen las características y las funcionalidades requeridas para la nueva versión del módulo que se construye. Una historia de usuario describe una funcionalidad que, por sí misma, aporta valor al usuario (18).

Las HU se componen de:

## Capítulo 2: Descripción de la Solución Propuesta

- Una descripción escrita de la historia usada como recordatorio y para planificar. (Debe ser breve).
- Conversaciones acerca de la historia que sirven para aclarar los detalles.
- Un criterio de aceptación (idealmente automatizado) que permita determinar cuándo la historia ha sido completada.

Las HU son dinámicas, ya que durante el desarrollo de la aplicación pueden cambiar constantemente el valor de prioridad de cada historia; así como su fecha de entrega. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto, el cual equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos (13).

### 2.5.1 Descripción de las historias de usuarios

Historia de usuario	
<b>Numero:</b> HU1	<b>Nombre:</b> Crear cilindro 3D
<b>Usuario:</b> Diseñador	
<b>Prioridad:</b> Alta	<b>Riesgo en desarrollo:</b> Alta
<b>Puntos estimados:</b> 2	<b>Iteración asignada:</b> 1
<b>Programador:</b> Roberto Menejías García	
<b>Descripción:</b> El usuario luego de haber introducido los valores del radio y la altura y seleccionado el punto central del sólido, podrá ser capaz de crear un cilindro en el visor 3D pulsando <i>Done</i> en el diálogo que se levanta como interfaz cuando se selecciona el icono del cilindro en el <i>ribbon</i> de la herramienta.	

## Capítulo 2: Descripción de la Solución Propuesta

**Observaciones:**

**Prototipo de interfaz:**

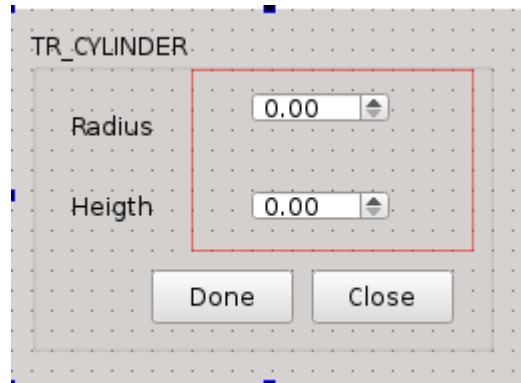


Tabla 6. HU1 Crear cilindro 3D

Historia de usuario	
<b>Numero:</b> HU2	<b>Nombre:</b> Crear esfera 3D
<b>Usuario:</b> Diseñador	
<b>Prioridad:</b> Alta	<b>Riesgo en desarrollo:</b> Alta
<b>Puntos estimados:</b> 2	<b>Iteración asignada:</b> 1
<b>Programador:</b> Roberto Menejías García	
<b>Descripción:</b> El usuario será capaz de construir una esfera en el visor 3D, seleccionando el punto central de la esfera y pasándole en forma de parámetro el valor del radio a la interfaz luego de haber seleccionado el icono del sólido en la barra de modelado 3D.	

## Capítulo 2: Descripción de la Solución Propuesta

**Observaciones:**

**Prototipo de interfaz:**

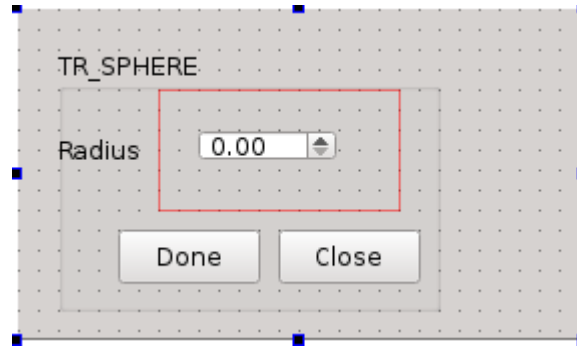


Tabla 7. HU2 Crear esfera 3D

Las restantes historias de usuario se pueden consultar en los anexos.

Definidos los requisitos funcionales y no funcionales del módulo, se describe a continuación la arquitectura que se utilizará para diseñar el sistema.

### 2.6 Arquitectura de la aplicación

Para el desarrollo de la solución se utilizará una arquitectura en 3 Capas, lo que permitirá que las distintas partes del módulo se puedan modificar sin tener que afectar la otra capa, es decir, separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, la lógica de negocio y el almacenamiento de los datos como tal.



## Capítulo 2: Descripción de la Solución Propuesta

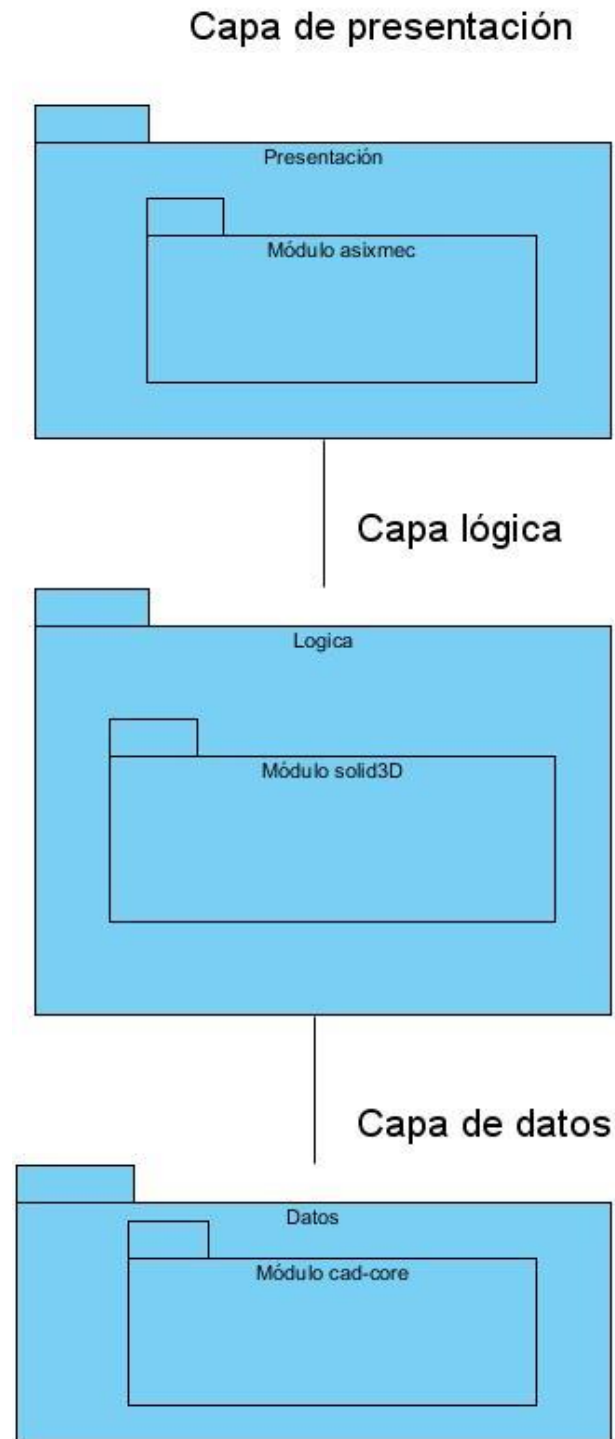


Ilustración 5. Modelo de la arquitectura del módulo.

## Capítulo 2: Descripción de la Solución Propuesta

---

**Capa de presentación:** se refiere a la parte del módulo que hace frente al usuario en la herramienta, esta presentación debe cumplir su propósito con el usuario final, de una interfaz amigable y fácil de usar (19). En la capa de presentación se encuentran clases que sirven para las operaciones de interfaz de usuario y los diálogos que se levantan en la creación de los sólidos. La capa utiliza el módulo Asixmec de la herramienta para desarrollar la solución propuesta.

**Capa lógica:** es donde se encuentran implementadas todas las clases, métodos y algoritmos que son ejecutados, recibe las peticiones del usuario y posteriormente envía las respuestas tras el proceso de creación de las primitivas. Esta capa es muy importante pues es donde se establecen todas aquellas reglas y restricciones que se tendrán que cumplir para la creación paramétrica de los sólidos Cilindro y Esfera. En esta capa esta como tal la lógica del negocio del módulo como propuesta de solución. (19)

**Capa de datos:** esta capa es la encargada de hacer las transacciones a el árbol OCAF de la herramienta, el cual esta generado y gestionado en el módulo *Cad-core* del sistema. En esta capa se almacenan todos los sólidos creados por el sistema, así como todos los elementos que los conforman haciéndolos persistentes. La consistencia de los datos ingresados o insertados es sumamente importante, los datos deben ser precisos. Esta capa mantendrá la comunicación con la de lógica de negocio al enviar la información que será procesada e ingresada en objetos según sea necesario. (19)

### 2.7 Conclusiones parciales

- En el capítulo se definieron los requisitos funcionales y no funcionales y se estableció la arquitectura que debe tener el módulo a desarrollar, la cual, permitira obtener ventajas significativas en su utilización.
- Los artefactos generados de la ingeniería sirven de base y referencia para la adición en un futuro de nuevos sólidos primitivos 3D al módulo.

### Capítulo 3. Características de la solución propuesta

En el presente capítulo se describe el proceso de implementación de la solución propuesta; brindando una descripción de las diferentes clases de la aplicación. Luego del análisis realizado y la problemática existente, se propone desarrollar un módulo que permita incorporar las funcionalidades de crear, de forma directa, sólidos primitivos básicos cilindros y esferas en el visor 3D de la herramienta, facilitando la interacción con el cliente y el mejor cumplimiento de las funcionalidades que se esperan ver cumplidas con el despliegue de la herramienta. La arquitectura del módulo será descrita con detalle a lo largo de este capítulo, fundamentando el papel que desempeña cada clase diseñada y el flujo que se sigue en las operaciones.

#### 3.1 Módulo solid3D

El módulo proveerá la funcionalidad de construir cilindros y esferas de forma directa en el visor 3D de la herramienta. Este proceso se ejecutará de tal forma que los sólidos creados generen, tanto el sólido como sus atributos en forma de parámetros. Estos elementos son construidos por las clases *Command*, y almacenados en forma de nodo en el árbol OCAF, utilizando algoritmos implementados con APIs (Interfaz de Programación Aplicada) de la biblioteca *OpenCascade*. Una vez almacenados los elementos en el árbol OCAF, las clases *Driver* de cada entidad 3D, dibujarán el sólido, haciendo uso de las clases y métodos, que ya se encuentran implementados en *AsiXMeC 1.0* con estos fines, extrayendo los elementos del nodo donde está almacenado el sólido, modificando su forma a partir del resultado obtenido y actualizando dicha información en el árbol de OCAF. Toda la lógica del proceso es gestionada por las clases *Controller*, las cuales monitorean, controlan y validan los parámetros y estados procedentes de las clases *Dialog* y *State* de cada primitiva. El *Controller* manda a ejecutar las operaciones de pre-visualizar, crear y por último visualizar los objetos, dado los estados y las validaciones necesarias en los parámetros de los cilindros y las esferas.

El proceso anteriormente descrito se muestra en la siguiente imagen con el objetivo de mejorar la comprensión del flujo de trabajo que sigue el sistema.

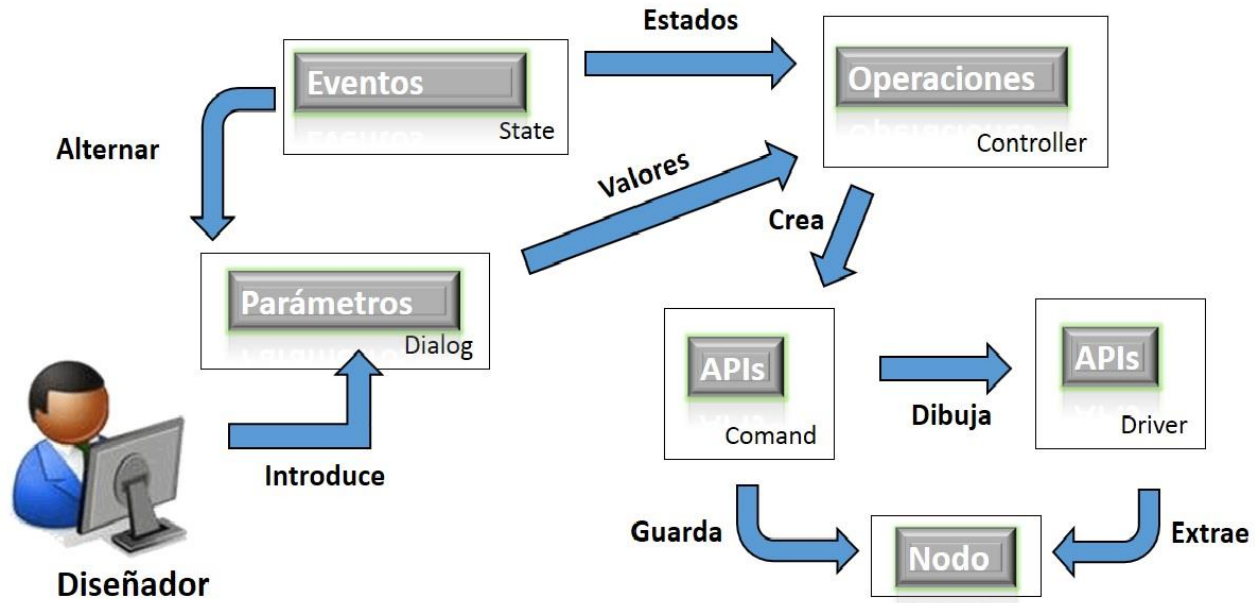


Ilustración 6. Flujo del sistema

A continuación, se describen con más detalles, las principales clases que intervienen en el flujo del sistema, con el fin de esclarecer el proceso de creación de las primitivas.

### 3.2 Diagrama de clases del diseño

Dentro del flujo de trabajo de diseño, la fase de elaboración de los diagramas de clases de diseño ocupa un importante papel, pues dichos diagramas muestran las clases finales del diseño con sus atributos y métodos y la forma en que se relacionan entre sí para la realización de los requisitos funcionales.

# Capítulo 3: Características del sistema

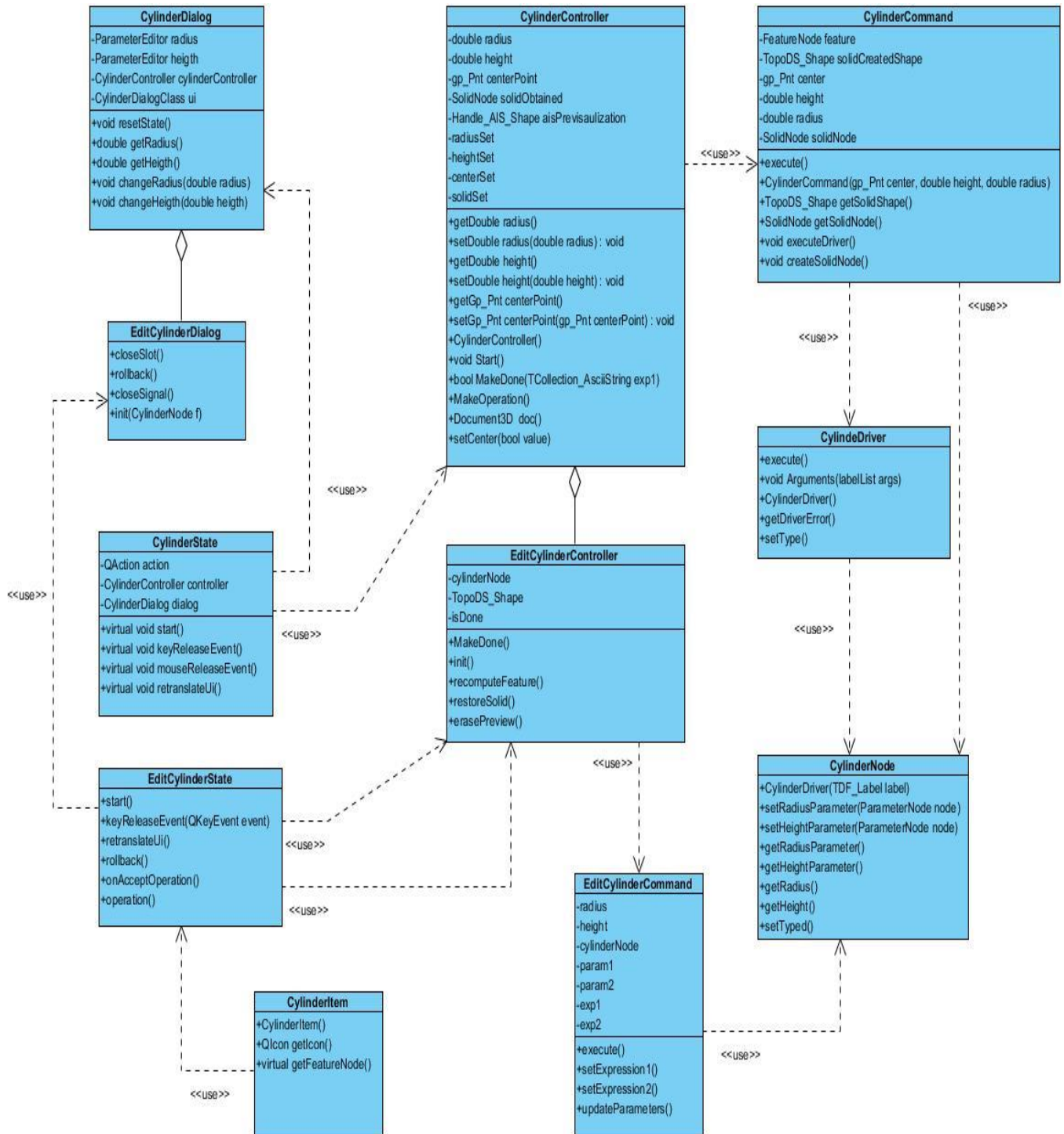


Ilustración 7 Diagrama de clases de cilindro

### 3.3 Descripción de las clases del diseño

Se describen las clases del sistema de forma específica, con sus atributos, funcionalidades y relaciones con el objetivo de explicar el flujo de trabajo que se sigue para crear paramétricamente los sólidos Cilindro y Esfera.

Nombre	Descripción	Ubicación
<b>CylinderDialog</b>	Es una interfaz que levanta el diálogo a través del cual el usuario seleccionará las dimensiones con las que va a crear el cilindro. Este diálogo es activado cuando es pulsado el icono de la primitiva en la interfaz principal de , dentro de la barra de modelado 3D de AsiXMeC 1.0.	Capa de presentación
<b>CylinderState</b>	Esta clase modifica los estados de la creación del objeto según los eventos del mouse en el visor 3D. Contiene un método por cada evento del mouse y cambia los valores del diálogo al cual está asociado. Es la clase intermediaria entre la capa de presentación y la lógica. Algunos de sus atributos principales son: <ul style="list-style-type: none"><li>• <b>cylinderController:</b> es una instancia de la clase Controller mediante la cual se llaman a los métodos de esa clase.</li><li>• <b>cylinderDialog:</b> instancia de la clase <i>CylinderDialog</i>.</li></ul>	Capa de presentación
<b>CylinderController</b>	Permite controlar los parámetros necesarios y validar que estén las condiciones necesarias para crear el cilindro. Contiene los métodos para gestionar todo el proceso de	Capa lógica

## Capítulo 3: Características del sistema

	<p>construcción y visualización del sólido. Sus principales atributos son:</p> <ul style="list-style-type: none"><li>• <b>radius:</b> es el valor que le da el usuario al radio del cilindro.</li><li>• <b>height:</b> es el valor que le da el usuario a la altura del cilindro.</li><li>• <b>centerPoint:</b> describe un punto en el espacio que será el centro de la figura.</li><li>• <b>solidObtained:</b> contiene el nodo del sólido generado, el cual será almacenado en el árbol OCAF.</li></ul>	
<b>CylinderCommand</b>	<p>Construye y guarda el cilindro en el árbol OCAF en tiempo de ejecución en forma de nodo, cuando es llamado por el <i>Controller</i>. Sus principales atributos son:</p> <ul style="list-style-type: none"><li>• <b>solidCreatedShape:</b> contiene el sólido creado.</li><li>• <b>feature:</b> contiene el nodo con los elementos que se utilizaron en la operación para construir el cilindro.</li><li>• <b>execute():</b> ejecuta la construcción del cilindro y su nodo.</li><li>• <b>getSolidNode():</b> retorna el nodo creado del cilindro.</li></ul>	Capa lógica

## Capítulo 3: Características del sistema

	<ul style="list-style-type: none"><li>• <b>getSolidShape():</b> retorna el <i>Shape</i>(forma) asociado al nodo del cilindro.</li></ul>	
<b>CylinderDriver</b>	<p>Brinda el algoritmo con las APIs de <i>OpenCascade</i>, para dibujar el cilindro y una vez este creado y almacenado en el árbol, extrayendo sus elementos del nodo del cilindro, para proyectarlo en el visor 3D. Sus principales funcionalidades son:</p> <ul style="list-style-type: none"><li>• <b>Arguments(TDF_LabelList):</b> extrae los datos del cilindro del árbol para ejecutar la proyección del sólido almacenado.</li><li>• <b>Execute(TFunction_Logbook):</b> ejecuta el dibujo y la proyección del cilindro cuando es llamado por el <i>execute()</i> del <i>Command</i>.</li></ul>	Capa lógica
<b>CylinderNode</b>	<p>Contiene el nodo del cilindro en el árbol OCAF, con todos sus elementos y características almacenadas en forma de nodo. Sus principales funcionalidades son:</p> <ul style="list-style-type: none"><li>• <b>CylinderNode ():</b> método que le da un <i>label</i> como identificador del nodo en el árbol.</li><li>• <b>setRadiusParameter():</b> modifica el valor del nodo del radio del cilindro en el árbol.</li><li>• <b>getRadiusParameter():</b> devuelve el nodo del parámetro radio del cilindro.</li></ul>	Capa de datos



## Capítulo 3: Características del sistema

	<ul style="list-style-type: none"><li>• <b>setType():</b> modifica el tipo de nodo al que pertenece la clase.</li></ul>	
<b>EditCylinderDialog</b>	<p>Esta clase es el diálogo que se levanta al seleccionar la opción de editar el cilindro creado. Es hija de <i>CylinderDialog</i>, por lo que hereda varias funcionalidades y atributos de la clase padre. Sus principales funcionalidades son:</p> <ul style="list-style-type: none"><li>• <b>init (CylinderNode f):</b> prepara el diálogo para realizar la edición del cilindro.</li><li>• <b>rollback ():</b> el método que se ejecuta cuando se cierra el diálogo.</li><li>• <b>closeSignal ():</b> señal emitida al pulsar el botón <i>Close</i>.</li><li>• <b>closeSlot ():</b> funcionalidad que se ejecuta al ser emitida la señal <i>closeSlot</i>.</li></ul>	Capa de presentación
<b>EditCylinderState</b>	<p>Permite alternar el comportamiento de la operación según cambie su estado por los eventos de mouse en el área de trabajo y el <i>EditCylinderState</i>. Sus principales funcionalidades son:</p> <ul style="list-style-type: none"><li>• <b>start ():</b> inicializa el estado para editar el sólido creado en el documento activo.</li><li>• <b>keyReleaseEvent():</b> tiene definido en su implementación qué hacer en caso del evento del mouse.</li></ul>	Capa de presentación

## Capítulo 3: Características del sistema

	<ul style="list-style-type: none"><li>• <b>onAcceptOperation():</b> actualiza el ítem en el navegador de objetos luego de haber editado el cilindro en el documento activo.</li><li>• <b>rollBack():</b> función que se ejecuta cuando se cierra el diálogo.</li></ul>	
<b>EditCylinderController</b>	<p>Controla toda la lógica para realizar la operación de editar un cilindro ya creado. Hereda de la clase <i>CylinderController</i> algunos atributos y métodos necesarios para la edición del sólido. Sus principales atributos y funcionalidades son:</p> <ul style="list-style-type: none"><li>• <b>cylinderNode:</b> instancia del nodo del cilindro almacenado en el árbol OCAF.</li><li>• <b>isDone:</b> es una variable booleana para verificar que el sólido este creado antes de editar.</li><li>• <b>MakeDone():</b> método booleano. Devuelve verdadero si la edición del cilindro se realizó con éxito.</li><li>• <b>init():</b> inicializa las condiciones necesarias para la edición del cilindro.</li><li>• <b>recomputeFeature():</b> re-calcula el nodo del cilindro creado una vez editado el sólido asociado.</li></ul>	Capa lógica

## Capítulo 3: Características del sistema

	<ul style="list-style-type: none"><li>• <b>restoreSolid():</b> restaura el sólido una vez editado sus elementos.</li><li>• <b>erasePreview():</b> limpia la edición del cilindro, dejando el sistema listo para otra operación.</li></ul>	
<b>EditCylinderCommand</b>	<p>Ejecuta la re-construcción del cilindro creado inicialmente. Crea el sólido con las nuevas dimensiones y lo almacena en el árbol OCAF. Algunos de su métodos y atributos más importantes son:</p> <ul style="list-style-type: none"><li>• <b>cylinderNode:</b> instancia del nodo del cilindro almacenado en el árbol OCAF.</li><li>• <b>radius:</b> valor real del radio del cilindro.</li><li>• <b>height:</b> valor real de la altura.</li><li>• <b>param1:</b> instancia de la clase encargada de gestionar los valores numéricos como parámetros. Está asociada a el valor del radio del cilindro.</li><li>• <b>param2:</b> instancia de la clase encargada de gestionar los valores numéricos como parámetros. Está asociada al valor de la altura.</li></ul>	Capa lógica

## Capítulo 3: Características del sistema

	<ul style="list-style-type: none"><li>• <b>exp1:</b> atributo que representa una expresión asociada a la variable altura.</li><li>• <b>exp2:</b> atributo que representa una expresión asociada a la variable altura.</li><li>• <b>execute():</b> ejecuta la reconstrucción del cilindro creado con las nuevas dimensiones en sus elementos.</li><li>• <b>setExpression1():</b> modifica la expresión asociada a la variable radio del cilindro.</li><li>• <b>setExpression2():</b> modifica la expresión asociada a la variable altura del cilindro.</li><li>• <b>updateParameters():</b> actualiza los valores de los elementos del cilindro que son modificados en la edición del sólido.</li></ul>	
<b>CylinderItem</b>	<p>Representa con un icono el cilindro creado en el navegador de objetos de la herramienta. Mediante esta clase se inician las operaciones de editar y borrar el sólido construido. Sus principales funcionalidades son:</p> <ul style="list-style-type: none"><li>• <b>CylinderItem():</b> construye el ítem en el navegador de objetos de la herramienta.</li><li>• <b>getIcon():</b> devuelve el icono del sólido visualizándolo en el navegador.</li></ul>	Capa de presentación

	<ul style="list-style-type: none"><li>• <b>getFeatureNode():</b> devuelve una instancia del nodo del cilindro.</li></ul>	
--	--	--

Tabla 8. Descripción de las clases del cilindro

Las restantes clases con sus atributos y funcionalidades se podrán ver en los anexos.

### 3.4 Patrones de diseño

A continuación, se describen los patrones de diseño que fueron utilizados para organizar y facilitar la implementación del módulo y sus clases.

#### 3.4.1 Patrones GRASP

**Controlador:** patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado. (20)

El uso de este patrón se ve evidenciando en las clases *Controller* de cada primitiva, como por ejemplo el *CylinderController*. A través del *CylinderController* se pueden facilitar las peticiones realizadas por los diseñadores para la creación del cilindro, y que tendrán algún impacto o requieran de una respuesta del módulo.

**Experto:** se utiliza con el objetivo de asignar una responsabilidad a la clase que posee toda la información necesaria para realizarla y desempeñarla (21). Es el patrón más utilizado debido a su esencia, por lo cual es empleado en casi todas las clases que contiene el módulo ejemplo *CylinderCommand* y *CylinderDriver*.

**Alta cohesión:** es una medida que determina cuán relacionadas y adecuadas están las responsabilidades de una clase, de manera que no realice un trabajo colosal; una clase con baja cohesión realiza un trabajo excesivo, haciéndola difícil de comprender, reutilizar y conservar. (22)

En el presente módulo se evidencia una alta cohesión funcional cuando las distintas funcionalidades de las clases diseñadas colaboran para producir algún comportamiento bien definido. El proceso se inicia desde la selección del usuario de la primitiva que desea construir,

hasta la creación y proyección del sólido en el visor 3D, funciona de manera armónica mediante la conjunción acoplada de las responsabilidades asignadas a cada clase.

**Bajo Acoplamiento:** es una medida de la fuerza con que una clase se relaciona con otras; es decir, no posee numerosas relaciones. (22)

El uso del patrón se puede ver evidenciado a través de la arquitectura en capas, donde estaría estructurada por capa de presentación, capa lógica y capa de datos, evitando que si existe algún cambio en clases de las capas inferiores no afectaría múltiples clases en las capas superiores solo a las relacionadas con ellas.

### 3.4.2 Patrones GOF

Se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro *Design Patterns Elements of Reusable Software* de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides. Los patrones *Gand of Four* (GOF) empleados en el desarrollo del sistema, son los dos que se describen a continuación.

**Command:** patrón de comportamiento muy sencillo y con muchas aplicaciones útiles, ya que ordena las acciones y tareas de una aplicación, definiendo una clase para cada tarea u operación que implemente una interfaz común (23). En el módulo, el patrón queda evidenciado en las clases *CylinderCommand* y *SphereCommand*.

**State:** permite a un objeto alterar su comportamiento cuando su estado interno cambia. Es recomendable su utilización cuando el comportamiento del objeto depende de su estado, y debe cambiar el comportamiento en tiempo de ejecución de acuerdo a ese estado (23). En el presente módulo, el uso de este patrón se evidencia en clases como *CylinderState* y *SphereState*, las cuales representan los estados de selección y creación de esas primitivas por los eventos del mouse, en dependencia del lugar dentro del visor 3D, donde sea pulsado.

### 3.5 Estándar de código

Un estándar de programación es: una forma de "normalizar" la programación de forma tal que, al trabajar en un proyecto, cualquier persona involucrada en el mismo tenga acceso y comprenda el código. (23)

#### Estilo de comentario

El estilo de los comentarios debe ser como el estilo de comentarios para C (`/* */` o `//`), no debe utilizarse el estilo de comentarios de Perl (`#`).

Ejemplo de comentario: `/*Flags*/`

#### Variables

Se declarará una variable por línea. Además, se evitará nombrar a las variables con abreviatura siempre que sea posible. Todos los caracteres serán escritos en minúscula, excepto en caso de que existan nombres compuestos. En este caso, se escribirá con minúscula el primer nombre y los restantes con letra inicial mayúscula, evitando usar el guión bajo. Ejemplo de esto: `bool radiusSet`

#### Métodos

Siguen el mismo convenio que las variables, los nombres de algunos métodos empiezan con un verbo. Cuando sean nombres compuestos, cada nombre empieza con mayúscula y se puede utilizar también, el guión bajo para separar los nombres.

Ejemplo de definición de métodos:

```
void setRadius (double r)
bool MakeDone(TCollection_AsciiString exp1)
void Start()
void MakeOperation()
```

#### Clases

Se declara una clase con la primera letra del nombre en mayúscula, si es un nombre compuesto el segundo nombre empieza con mayúscula, cuando se necesite un tercer nombre se le pone guión bajo entre los nombres.

Ejemplo de una clase:

```
class CylinderDriver: public ObjectDriver
```

```
{
public:
    CylinderDriver();

    /**
     * @brief Return the id of the class
     * @details
     * @return The id
     * @param
     */
    //static const Standard_GUID getId();

    void Arguments(TDF_LabelList &args) const;

    Standard_Integer Execute(TFunction_Logbook &log) const;

    /**
     * @brief getDriverError
     * @param driverResult
     * @return
     */
    string getDriverError(Standard_Integer driverResult);

    virtual ~CylinderDriver();

    DEFINE_STANDARD_RTTI(CylinderDriver)
}
```

### 3.6 Algoritmos de la solución

Para llevar a cabo la solución que se propone, tal y como se ha descrito, se hizo uso de la biblioteca *OpenCascade*, y de ella fueron utilizadas varias clases cuyas funcionalidades se describen a continuación:

- **TopoDS\_Shape**: define las entidades que son creadas en segunda y tercera dimensión en la herramienta.
- **AIS\_Shape**: define un marco para gestionar la presentación y selección de formas en la aplicación.



- **BRepPrimAPI\_MakeCylinder:** ejecuta los algoritmos para calcular topológicamente el cilindro resultante.
- **BRepPrimAPI\_MakeSphere:** ejecuta el algoritmo para calcular la esfera resultante.
- **TCollection\_AsciiString:** esta clase trabaja con las expresiones asociadas a los valores de cada elemento o dato que pueda tomar como referencia las variables en los cilindros y esferas.
- **Standard\_Real:** esta clase define las variables reales con los valores de los elementos de cada primitiva.
- **gp\_Pln:** define un plano en tercera dimensión.
- **gp\_Pnt:** define un punto 3D en un plano.

### 3.6.1 MakeDone

Este algoritmo implementado en el *Controller*, se encarga de llamar al *Command* para ejecutar la construcción del nodo y la creación de una referencia al sólido mediante el método *execute* (). Después de ser creado, el sólido es almacenado en tiempo de ejecución en el árbol OCAF con el nodo del objeto creado.

### Código del MakeDone

```
bool CylinderController::MakeDone(TCollection_AsciiString exp1) // Ejecuta el comando
{
    bool result = validateBeforeExecution();

    if(result)
    {
        clearPrevisualization();

        CylinderCommand* cmd;
        ((Selection3D*)doc()->getActiveView()->getSelection()->clearSelected();
        doc()->updateActiveViewer();
        doc()->getActiveView()->getSelection()->deactivateLocalSelection();

        doc()->newCommand();

        try
        {
            cmd = new CylinderCommand(centerPoint,height,radius);
            cmd->execute();
        }
        catch(...)
        {
            doc()->abortCommand();
            result = false;
        }

        doc()->commitCommand();

        delete cmd;
        Start();
    }

    return result;
}
```

Ilustración 8. Código del método MakeDone ().

### 3.7 Conclusiones parciales

- La arquitectura definida para el diseño e implementación del módulo garantiza su fácil extensibilidad, de modo que si se desean agregar nuevos sólidos basta con ajustarse a la misma, garantizando además la perfecta integración con el resto de los módulos de AsiXMeC 1.0.

## Capítulo 4: Pruebas del sistema

### 4.1 Introducción

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa. Involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados, es por eso que la realización de las mismas al software es un factor de vital importancia.

### 4.2 Pruebas

Las pruebas de software, son los procesos que permiten verificar y validar la calidad de un sistema. Son utilizadas para identificar posibles defectos en los productos desarrollados. Se integran dentro de las diferentes fases del ciclo del software que se ubican dentro de la Ingeniería de Software. (24)

Para determinar el nivel de factibilidad y calidad se le realizan, pruebas al módulo mediante técnicas experimentales usando pruebas de caja negra.

#### Pruebas de caja negra

- Pruebas que se llevan a cabo sobre la interfaz del módulo.
- El objetivo es demostrar que las funciones del módulo son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto (no se ve el código).

#### Técnica de partición y equivalencia

La partición de equivalencia es un método de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar [26].

### 4.3 Casos de pruebas

Son un conjunto de condiciones o variables mediante los cuales se determina si los requisitos de una aplicación son parcial o completamente satisfechos. Un caso de prueba consta de una entrada conocida y una salida esperada, estos son formulados antes de que se ejecute la prueba, donde

## Capítulo 4: Pruebas del sistema

la entrada conocida debe probar una precondition y la salida esperada debe probar una poscondición. Los valores asignados a las variables significan **V** (aparece y es válida), y **NA** (No aparece la variable o no es necesario el dato) y **I** (aparece, pero es invalida).

Escenario	Descripción	Radio	Altura	Respuesta del sistema	Flujo central
EC 1.1 Introducir las variables correctamente	El usuario introduce los valores de las variables necesarias correctamente	V 12	V 24	Esperada: el sistema construye el cilindro con las dimensiones introducidas.	<ol style="list-style-type: none"> <li>1. Pulsar Icono ``CYLINDER``</li> <li>2. Seleccionar el punto central del cilindro en entorno de trabajo 3D.</li> <li>3. Seleccionar el radio ``Radius``</li> <li>4. Seleccionar la altura ``Height``.</li> </ol> <p>3. Pulsar botón ``Done``.</p>
EC 1.2 Ausencia de algunas de las variables	Falta el valor de algunas de las variables necesarias	V 4	NA --	Esperada: el sistema no realiza ninguna operación al pulsar el botón ``Done``. Muestra un mensaje de advertencia.	<ol style="list-style-type: none"> <li>1. Icono ``CYLINDER``</li> <li>2. Seleccionar el punto central del cilindro en entorno de trabajo 3D.</li> <li>3. Seleccionar solo algunas de las variables necesarias</li> <li>4. Pulsar botón ``Done``.</li> </ol>
		NA --	V 36		
		NA --	NA --		
EC 1.3 Introducir valores inválidos	El usuario les dará valores inválidos a las	I as	I r	Esperada: el sistema no realiza ninguna	<ol style="list-style-type: none"> <li>1. Icono ``CYLINDER``</li> </ol>

## Capítulo 4: Pruebas del sistema

	variables del cilindro.	V 12.5	I -43	operación y emite un mensaje mostrando el error.	2. Seleccionar el punto central del cilindro en entorno de trabajo 3D.  3. Seleccionar el radio ``Radius``  4. Seleccionar la altura ``Height``.  5. Pulsar botón ``Done``.
--	-------------------------	-----------	----------	--	---

Tabla 9. Caso de pruebas del HU1 (tabla 1 de 2)

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Radio	verticalLayout	No	Contiene el valor del radio del cilindro. Tiene un rango mayor que 0 y menor que 1000. No permite valores negativos.
2	Altura	verticalLayout	No	Contiene el valor de la altura del cilindro. Tiene un rango mayor que 0 y menor que 1000. No permite valores negativos.

Tabla 10. Caso de pruebas del HU1 (tabla 2 de 2) (Descripción de las variables del Caso de Prueba)

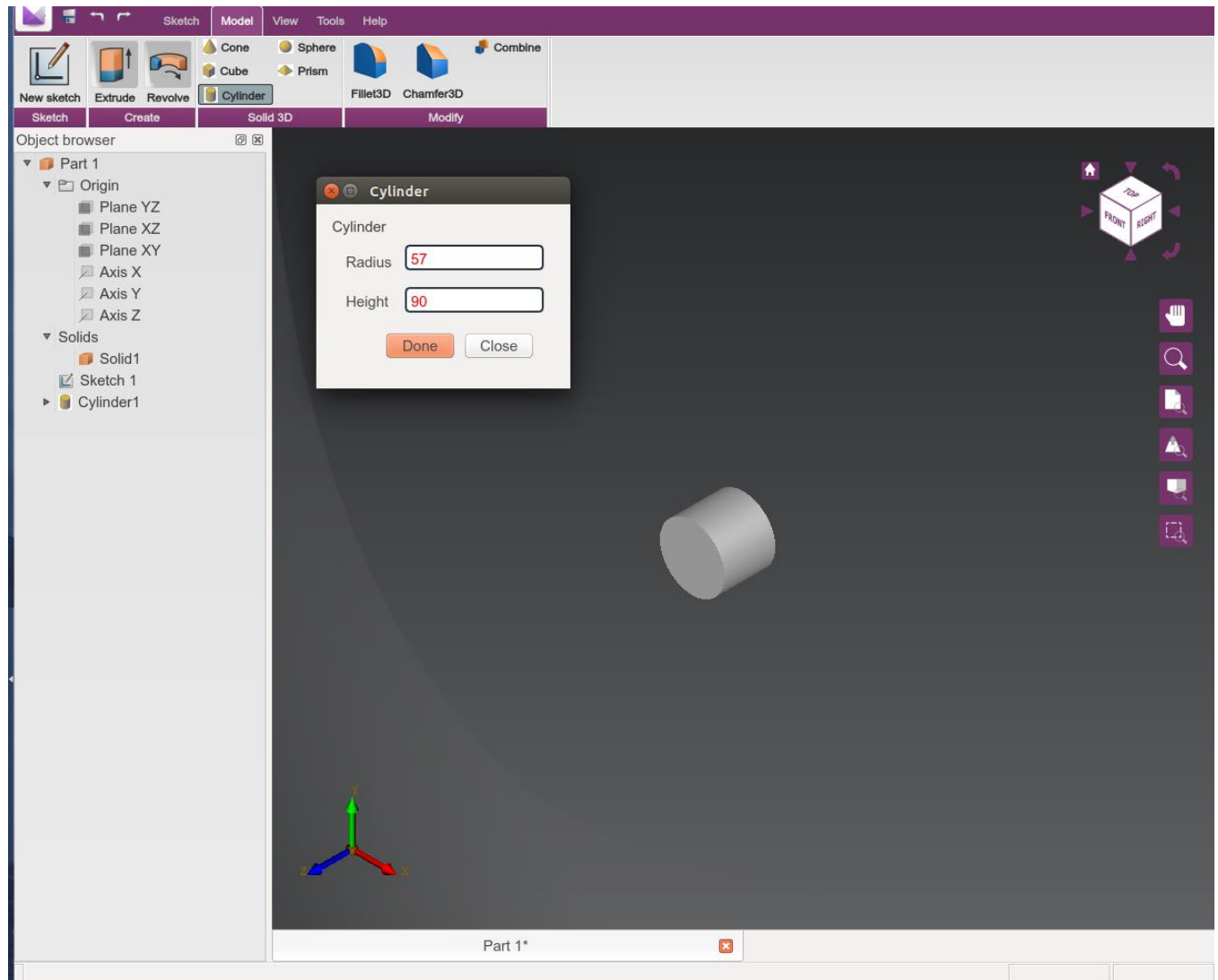
Los restantes casos de pruebas a las demás HU se pueden ver en los anexos.

### 4.4 Resultados de las pruebas

Teniendo en cuenta las condiciones de la solución propuesta para la creación paramétrica y directa de los sólidos primitivos básicos (cilindro y esfera), se realizaron pruebas al módulo de las cuales se exponen sus resultados.

#### Pruebas de funcionalidad:

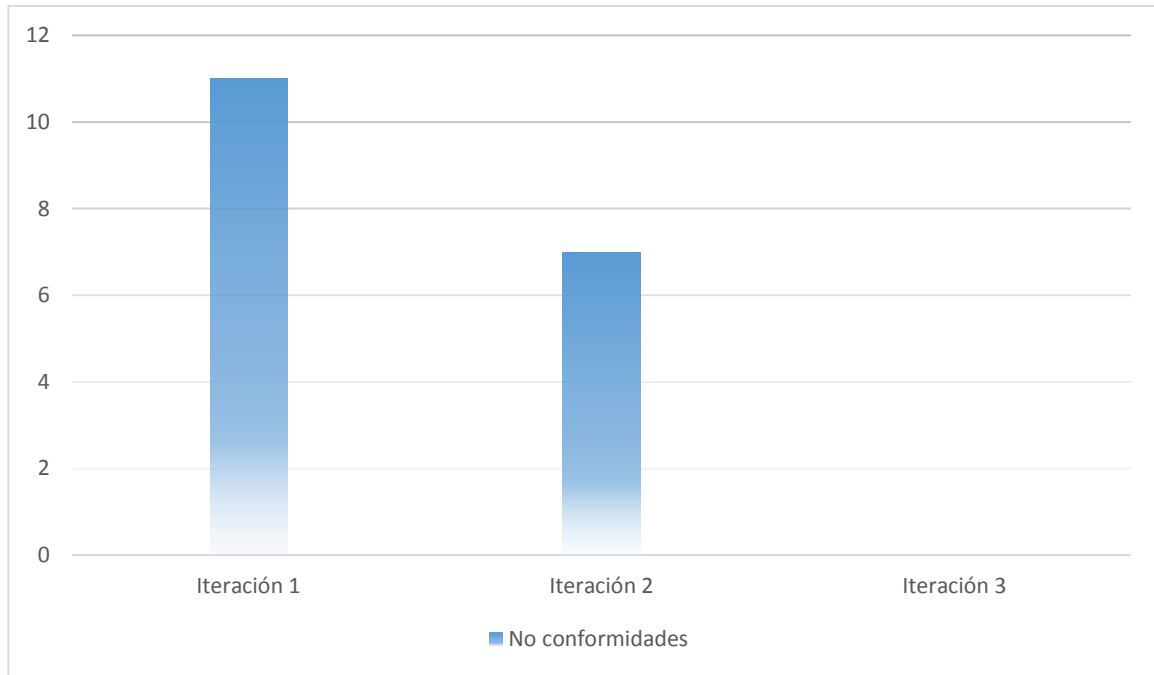
Se aplicaron pruebas de caja negra utilizando la técnica de partición y equivalencia con el objetivo de examinar los valores válidos e inválidos de las entradas existentes al sistema. Estas pruebas son las pruebas definidas en el proyecto por el equipo de desarrollo de AsiXMeC 1.0 para verificar la especificación de los requerimientos funcionales del sistema.



**Ilustración 9. Crear cilindro**

Una vez diseñados los 10 casos de pruebas se procedió a la ejecución de los mismos. Los resultados se pueden observar en la Ilustración 11, donde se muestra la cantidad de no conformidades que posteriormente fueron corregidas. Se realizaron 3 iteraciones, de las cuales las dos primeras resultaron con deficiencias y para la tercera se erradicaron las mismas. En la primera iteración se detectaron 11 no conformidades, en la segunda 7 y en la tercera no se detectó alguna.

La siguiente ilustración muestra gráficamente los resultados de las pruebas de caja negra por iteración.



**Ilustración 10. No conformidades por iteración.**

### **Pruebas de aceptación:**

Se realizó un encuentro con algunos miembros del personal del proyecto DISEM, los cuales tienen roles de desarrolladores y analistas, con el objetivo de que pudieran utilizar el sistema totalmente funcional. Luego de explicado el funcionamiento del módulo y la interacción de los especialistas con este, se pasa a medir el nivel de satisfacción al grupo de los 5 profesionales consultados. Como resultado, 2 de los miembros mostraron un nivel alto de satisfacción, 2 medio, y 1 presentó un bajo agrado por la propuesta de solución. La siguiente ilustración muestra los resultados.

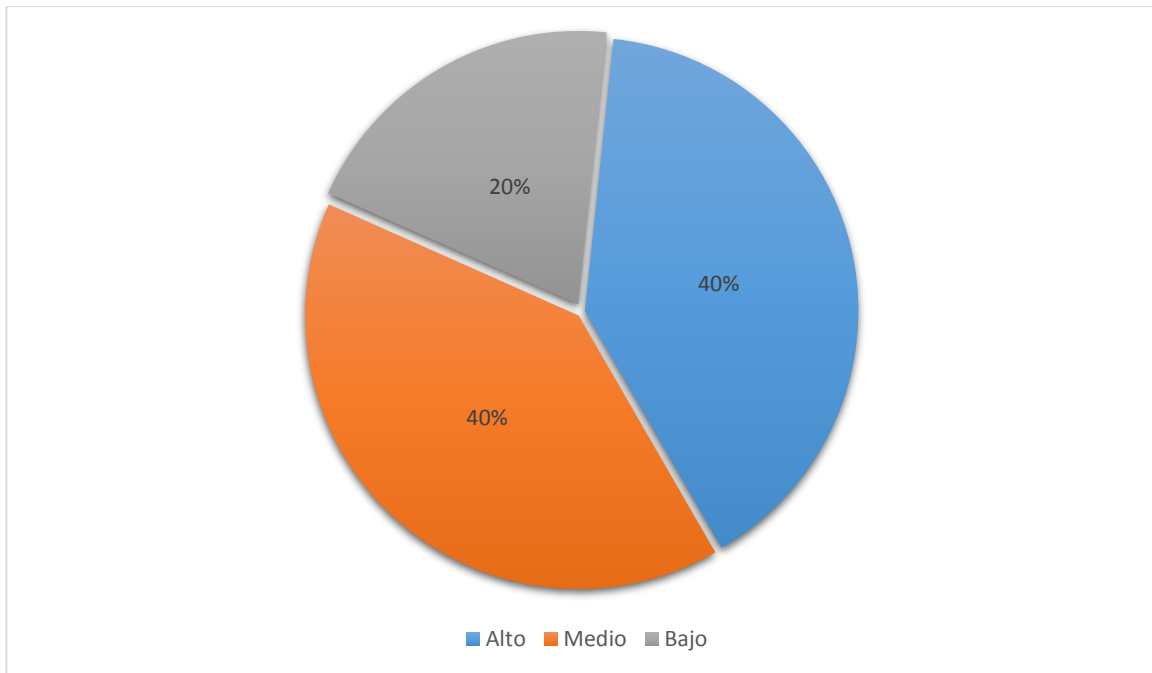


Ilustración 11. Nivel de aceptación.

### 4.5 Conclusiones parciales

- Se validó la solución mediante el uso de pruebas de aceptación, midiendo el nivel de satisfacción del cliente con la propuesta, el cual mantuvo un criterio alto respecto a la solución, verificando que el sistema está listo para ser usado.
- Se le aplicó pruebas de caja negra al sistema, validando el resultado de la implementación y el funcionamiento del módulo de acuerdo con las especificaciones de requisitos y funciones para las cuales fue construido.



## Conclusiones Generales

- La arquitectura definida en la herramienta AsiXMeC posee deficiencias que no permiten la creación de entidades básicas a partir de las llamadas a sus comandos desde el visor 3D, pues no existe una forma de conocer el *sketcher* 2D que está asociado al visor 3D activo.
- La inserción de un módulo que cree de forma directa sólidos en el visor 3D de AsiXMeC, soluciona problemas de diseño de piezas que hasta el momento no se podían realizar, como por ejemplo todas aquellas que tengan como base una esfera.

### Recomendaciones

- Incorporar al módulo la creación de los demás sólidos primitivos 3D: pirámide, cono y cubo, para garantizar de esta forma la realización de cualquier tipo de diseño.
- Implementar dentro de la arquitectura de AsiXMeC un mecanismo que permita conocer el *sketcher* asociado a un visor 3D, de modo que se puedan crear desde el 3D puntos y entidades básicas 2D que intrínsecamente se construyan sobre dicho *sketcher*, lo que permitiría generar los bocetos 2D que sirven como base a cada sólido y modificarlos.

## Referencias Bibliográficas

1. Scribd. *Sistemas-CAD*. [En línea] [Citado el: 21 de Mayo de 2016.] <http://es.scribd.com>.
2. Wasim, Younis. *Inventor y su simulación con ejercicios*. s.l. : S.A. MARCOMBO, 2012.
3. *Técnicas de Representación Gráfica AutoCAD CAD 3D*. s.l. : Universidad de Cantabria, 2010-2011.
4. Foley, James D. Van Dan, Andries. INTRODUCCION A LA GRAFICACION POR COMPUTADOR. *GRAFICOS POR COMPUTADORA*. [En línea] [Citado el: 14 de junio de 2016.] <http://hdl.handle.net/123456789/562>.
5. *Métodos "I+D" de la Informática, Graciela Elisa Barchini*. Santiago del Estero, Argentina : Universidad Nacional de Santiago del Estero , 1912.
6. Blanco Fernadez, Julio y Sanz Adan, Felix. *CAD-CAM: Graficos, Animación y Simulación*. s.l. : S.A. Ediciones Paraninfo, 2002.
7. Barana Caparros, Francisco. *Como modelar con AutoDesk Inventor 2014*. s.l. : RA-MA, 2015.
8. Estrada, Ing. Jose Angel Lores. *Módulo sweep para la herramienta Asixmec*. Habana : s.n., 2013.
9. Mixprogramas. [En línea] [Citado el: 22 de Mayo de 2016.] <http://www.mixprogramas.com>.
10. Rojas, Alan D. Osorio. *Qt 4 Manual Introductorio*. 16 de enero 2008.
11. Stroustrup, Bjarne. *The Handobook of Object Technoly*. 1999.
12. Nieves, I.I.S. *METODOLOGÍAS ÁGILES. Proceso Unificado Ágil (AUP). Ingeniería del Software II – Análisis de Sistemas*, . 2014.
13. Rodríguez Corbea, Maite y Meylin Ordoñez Pérez. *La metodología XP aplicable al desarrollo del software educativo en Cuba*. Habana, Cuba : Universidad de las Ciencias Informáticas, 2007.
14. *Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información*. González, L.C. and E.R.P. Torres. Habana, Cuba : Serie Científica de la Universidad de las Ciencias Informáticas, 2012.
15. Pugh, Anthony. *Polyhedra: A visual approach*. California : s.n., 1976.
16. buenastareas.com. [En línea] 2014. [Citado el: 12 de febrero de 2016.] <http://www.buenastareas.com/ensayos/Modelos-Conceptuales/23455.html>.
17. Somerville, Ian. *Ingeniería del software*. s.l. : Pearson Educación, 2005.

## Bibliografía Referenciada

---

18. Beas, J.M. Historias de Usuario. [En línea] 2011. [http://jmbeas.es/guias/historias-de-usuario/..](http://jmbeas.es/guias/historias-de-usuario/)
19. Zamora, C.E.R. *La programación por capas MilkZoft*. s.l. : CODEJOBS Aprende a programar., 2014.
20. Carmona, J.G. *in GRASP: Controlador*. s.l. : Juan García Carmona, 2012.
21. Rey, E.M. *Diseño Dirigido por Responsabilidades con los patrones GRASP*. . s.l. : Pearson Educación, S.A., 2015.
22. Tabares, R.B. *Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación*. [http://biblioteca.ucp.edu.co/ojs/index.php/entrecei/article/view/760/721] 2011.
23. Larman, Craig. *UML y Patrones: Introducción al análisis y diseño orientado a objetos y proceso unificado*. . Madrid. : Segunda edición, 2003.
24. Juristo, Natalia y Vegas, Sira. Técnicas de Evaluación de Software. [En línea] A.M.M, 2006. [Citado el: 23 de Junio de 2016.] <http://grise.upm.es>.
25. Fowler, Martin y Scott, Kendall. *UML gota a gota*. 1999.
26. Sánchez, T.R. *PROGRAMA DE MEJORA Metodología de desarrollo para la Actividad productiva de la UCI*. Habana, Cuba : s.n., 2014.
27. SADRADÍN, L.C. *SISTEMA AUTOMATIZADO PARA EL DISEÑO DE PROTOTIPOS DE INTERFACES DE USUARIO*. s.l. : Universidad de las Ciencias Informáticas, 2009.
28. Cham, F. *Introducción al diseño arquitectónico*. . 2013.

## Anexos

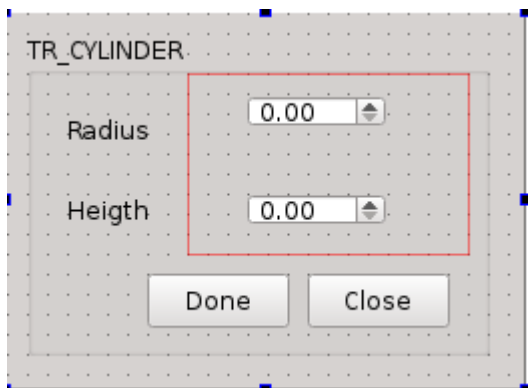
Historia de usuario	
<b>Numero:</b> HU3	<b>Nombre:</b> Editar cilindro 3D
<b>Usuario:</b> Diseñador	
<b>Prioridad:</b> Baja	<b>Riesgo en desarrollo:</b> Alta
<b>Puntos estimados:</b> 2	<b>Iteración asignada:</b> 1
<b>Programador:</b> Roberto Menejías García	
<b>Descripción:</b> Una vez construido el cilindro, el usuario puede modificar a conveniencia, las dimensiones del radio y la altura del sólido. Se reconstruye el cilindro con las dimensiones nuevas en sus elementos.	
<b>Observaciones:</b>	
<b>Prototipo de interfaz:</b>	
	

Tabla 11. HU3 Editar cilindro 3D

Historia de usuario	
<b>Numero:</b> HU4	<b>Nombre:</b> Editar esfera 3D
<b>Usuario:</b> Diseñador	
<b>Prioridad:</b> Baja	<b>Riesgo en desarrollo:</b> Alta
<b>Puntos estimados:</b> 2	<b>Iteración asignada:</b> 1
<b>Programador:</b> Roberto Menejías García	

**Descripción:** Permite que el usuario modifique la esfera cuando está creada en el visor 3D. Las modificaciones realizadas a el valor del radio hacen que se reconstruya la esfera con esa nueva dimensión.

**Observaciones:**

**Prototipo de interfaz:**

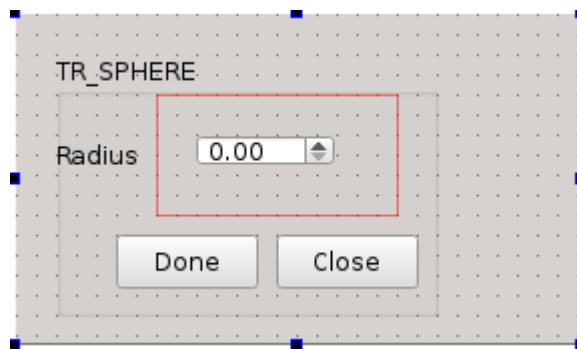


Tabla 12. HU4 Editar esfera 3D

Historia de usuario	
<b>Numero:</b> HU5	<b>Nombre:</b> Borrar cilindro 3D
<b>Usuario:</b> Diseñador	
<b>Prioridad:</b> Alta	<b>Riesgo en desarrollo:</b> Alta
<b>Puntos estimados:</b> 2	<b>Iteración asignada:</b> 1
<b>Programador:</b> Roberto Menejías García	
<b>Descripción:</b> Permite borrar el cilindro creado del área de trabajo. Destruye el ítem que representa el sólido en el navegador de objetos y elimina el nodo del cilindro en el árbol OCAF.	
<b>Observaciones:</b> La operación se inicia al seleccionar la opción de borrar al desplegarse, dando <i>click</i> izquierdo en el ítem del sólido.	

Tabla 13. HU5 Borrar cilindro 3D

Historia de usuario	
<b>Numero:</b> HU6	<b>Nombre:</b> Borrar esfera 3D

<b>Usuario:</b> Diseñador	
<b>Prioridad:</b> Alta	<b>Riesgo en desarrollo:</b> Alta
<b>Puntos estimados:</b> 2	<b>Iteración asignada:</b> 1
<b>Programador:</b> Roberto Menejías García	
<b>Descripción:</b> Permite borrar la esfera creada del área de trabajo. Destruye el ítem que representa el sólido en el navegador de objetos y elimina el nodo de la esfera en el árbol OCAF.	
<b>Observaciones:</b> La operación se inicia al seleccionar la opción de borrar al desplegarse, dando <i>click</i> izquierdo en el ítem de la primitiva.	

Tabla 14. HU6 Borrar esfera 3D

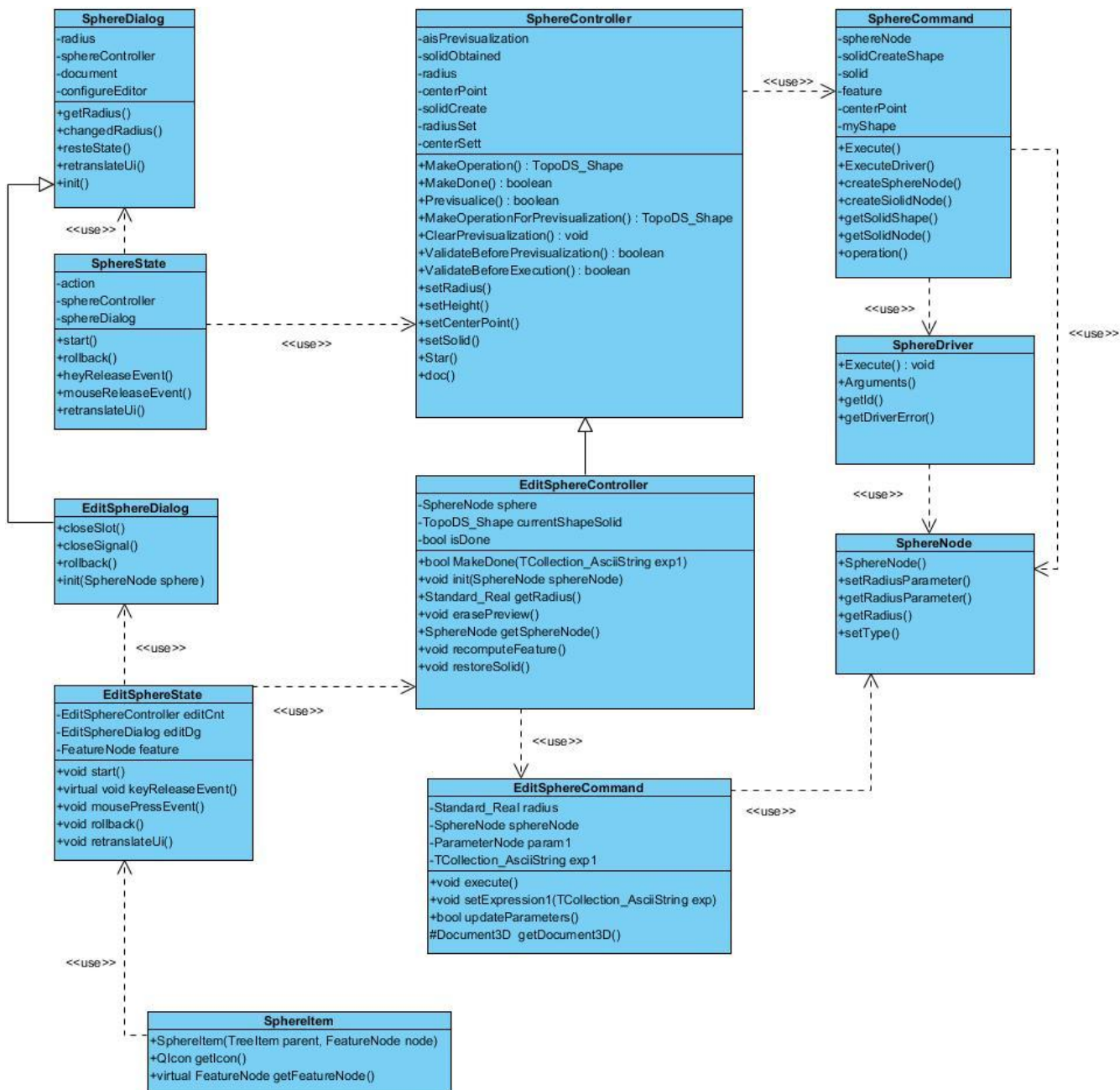


Ilustración 12. Diagrama de clases de la esfera



Nombre	Descripción	Ubicación
<b>SphereDialog</b>	Es una pequeña interfaz que levanta el diálogo por el cual el usuario seleccionará las dimensiones con las que va a crear la esfera. Este diálogo es activado cuando es pulsado el icono de la primitiva en la interfaz principal de AsiXMeC 1.0.	Capa de presentación
<b>SphereState</b>	Esta clase controla y modifica los estados del objeto según los eventos del mouse en visor 3D. Contiene un método por evento del mouse y cambia los valores del diálogo al cual está asociado. Es la clase intermediaria entre la capa de presentación y la capa lógica.	Capa de presentación
<b>SphereController</b>	Permite controlar los parámetros necesarios para construir la esfera. Contiene los métodos para gestionar todo el proceso de construcción y modificación de la esfera.	Capa lógica
<b>SphereCommand</b>	Genera el sólido utilizando la tecnología <i>OpenCascade</i> y los parámetros entrados. Guarda el nodo de la esfera en el árbol <i>OpenCascade</i> en tiempo de ejecución, cuando es llamado por el <i>Controller</i> .	Capa lógica
<b>SphereDriver</b>	Brinda el algoritmo con las APIs de <i>OpenCascade</i> , para dibujar la esfera una vez esta creada, visualizarla en el visor 3D. Toda	Capa lógica

	la operación la realiza utilizando los elementos extraídos del nodo de la esfera en el árbol OCAF.	
<b>SphereNode</b>	Es la instancia del árbol OCAF en la que se almacena la información referente a las características de la esfera. Esta clase hace persistente la esfera creada en ejecución.	Capa de datos
<b>EditSphereDialog</b>	Representa al diálogo que se levanta una vez seleccionada la acción de editar la esfera. Es una clase hija de <i>SphereDialog</i> .	Capa de presentación
<b>EditSphereState</b>	Permite alternar el estado de la operación editar, según los eventos de mouse. Conecta las señales emitidas del <i>EditDialogSphere</i> con sus <i>slots</i> .	Capa de presentación
<b>EditSphereController</b>	Controla toda la lógica para realizar la operación de editar una esfera ya creada. Hereda de la clase <i>SphereController</i> algunos atributos y métodos necesarios para la edición del sólido.	Capa lógica
<b>EditSphereCommand</b>	Ejecuta la re-construcción de la esfera creada inicialmente. Crea el sólido con las nuevas dimensiones y lo almacena en el árbol OCAF. Utiliza las APIs de <i>OpenCascade</i> .	Capa lógica
<b>SphereItem</b>	Representa con un icono la esfera creada en el navegador de objetos de la herramienta. Mediante esta clase se inician	Capa de presentación

	las operaciones de editar y borrar el sólido construido.	
--	--	--

Tabla 15. Descripción de las clases de diseño de la esfera

Escenario	Descripción	Radio	Altura	Respuesta del sistema	Flujo central
EC 1.1 Modificar algunas de las variables	El usuario modifica algunos de los valores de las variables necesarias	V NA ---	NA V 128 V 67	Esperada: el sistema modifica el cilindro creado	1. Pulsar Icono ``CYLINDER`` del <i>Item</i> . 2. Seleccionar la opción Editar. 3. Modificar los valores de las variables o alguna de ella 4. Pulsar botón ``Done``

Tabla 16. Caso de prueba para la HU3

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Radio	verticalLayout	No	Contiene el valor del radio del cilindro. Tiene un rango mayor que 0 y menor que 1000. No permite valores negativos.
2	Altura	verticalLayout	No	Contiene el valor de la altura del cilindro. Tiene un rango mayor que 0 y menor que 1000. No permite valores negativos.

Tabla 17. Caso de prueba para la HU3(Descripción de variables)

Escenario	Descripción	Radio	Respuesta del sistema	Flujo central
EC 1.1 Introducir la variable correctamente	El usuario introduce de forma correcta el valor de la variable necesaria	V 23	Esperada: el sistema crea la esfera con el valor del radio introducido	1. Pulsar Icono ``SPHERE`` 2. Seleccionar el punto central de la esfera en el área de trabajo 3D. 3. Seleccionar el radio ``Radius``

				3. Pulsar botón ``Done``
EC 1.2 Ausencia de la variable	El usuario no introduce valor a la variable necesaria	NA --	Esperada: el sistema no realiza ninguna operación al pulsar ``Done``	1. Icono ``SPHERE`` 2. Seleccionar el punto central de la esfera en el área de trabajo 3D. 3. No seleccionar valor para el radio 4. Pulsar botón ``Done``

Tabla 18. Caso de prueba para la HU2

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Radio	verticalLayout	No	Contiene el valor del radio del cilindro. Tiene un rango mayor que 0 y menor que 1000. No permite valores negativos.

Tabla 19. Caso de prueba para la HU2(Descripción de variable)

Escenario	Descripción	Radio	Respuesta del sistema	Flujo central
EC 1.1 Modificar valor de la variable	El usuario modifica el valor de la variable necesaria una vez creada la esfera	V 13	Esperada: el sistema modifica la esfera creada	1. Icono ``SPHERE`` del <i>Item</i> de la esfera. 2. Seleccionar la opción Editar. 3. Modificar el radio ``Radius`` 4. Pulsar botón ``Done``

Tabla 20. Caso de prueba para la HU4

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Radio	verticalLayout	No	Contiene el valor del radio del cilindro. Tiene un rango mayor que 0 y menor que 1000. No permite valores negativos.

Tabla 21. Caso de prueba para la HU4(Descripción de la variable)

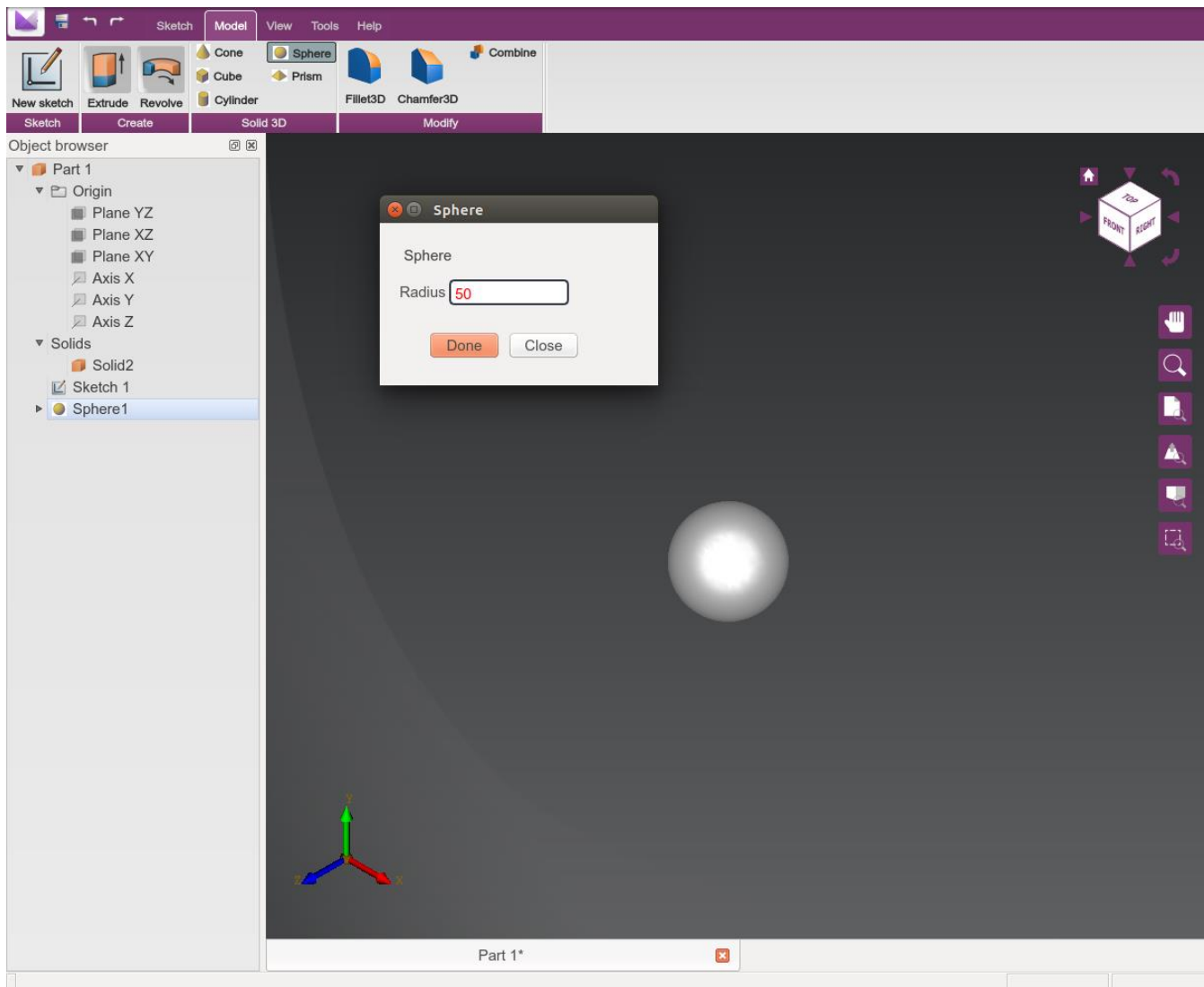


Ilustración 13. Crear esfera 3D

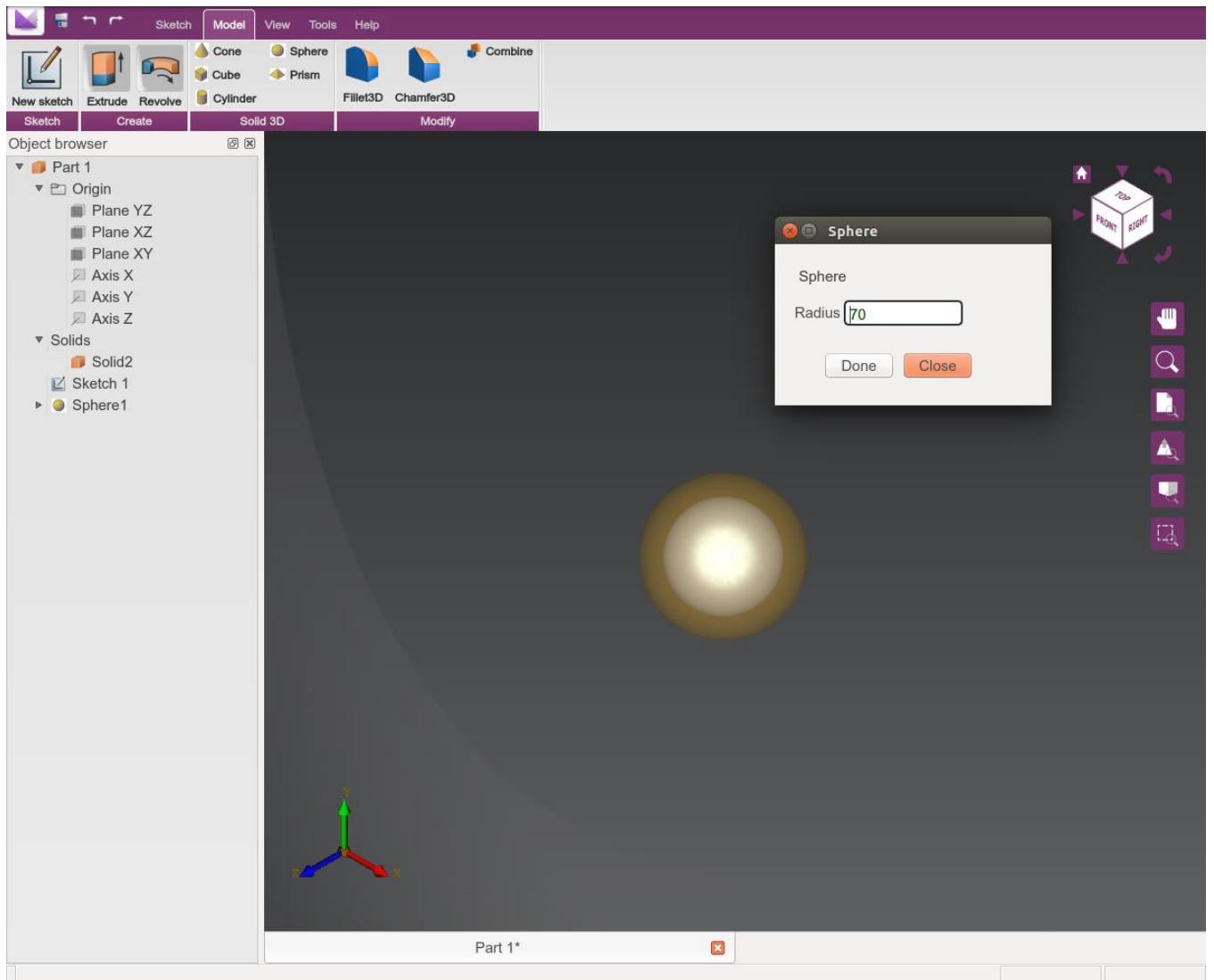


Ilustración 14. Editar esfera 3D

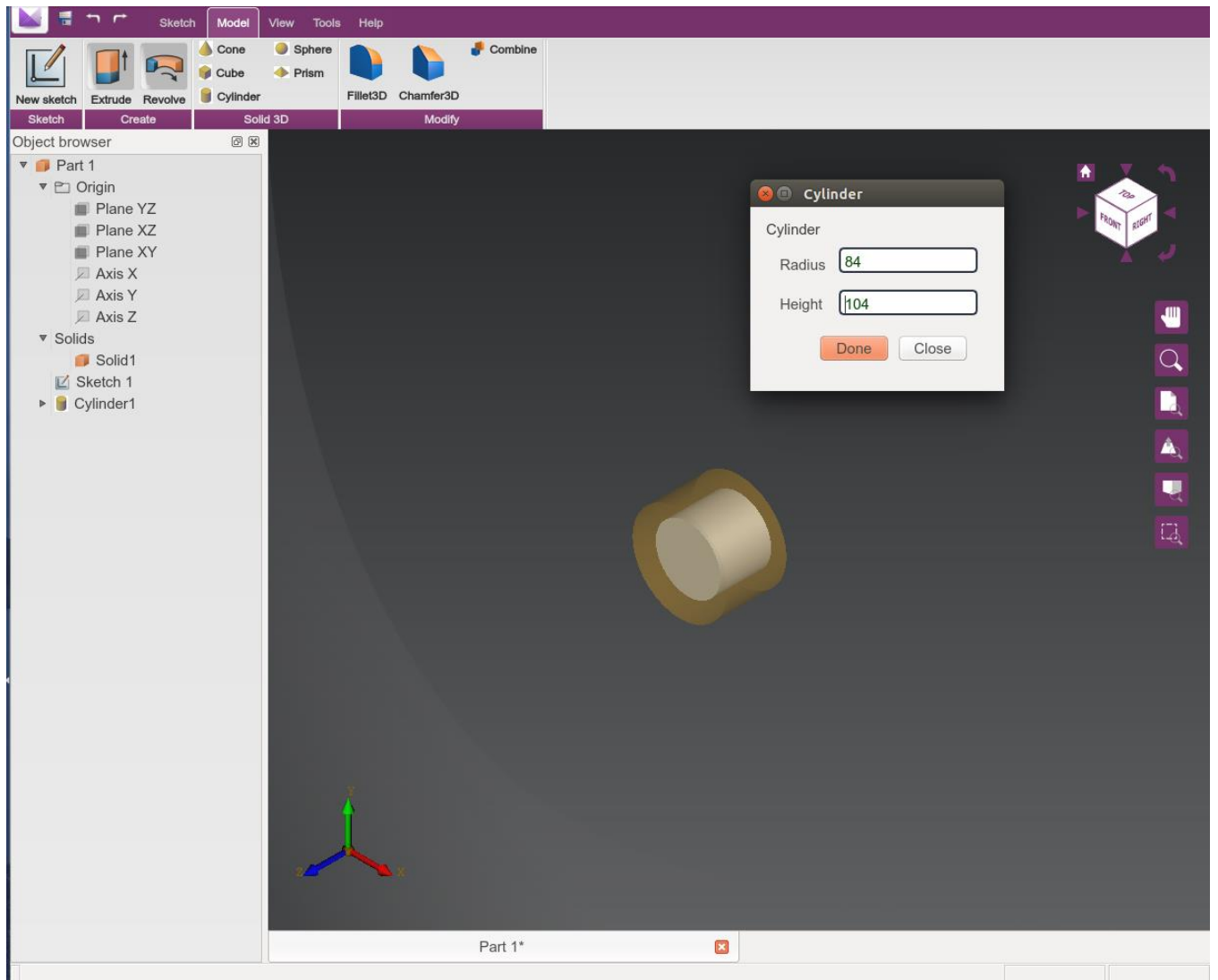


Ilustración 15. Editar cilindro 3D