UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 4

Vista de detalle para el módulo de planos técnicos del sistema AsiXmec

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

Autor: Arian Martínez Camejo

Tutores:

Dr. C. Hassán Lombera Rodríguez

M. Sc. Ángel Alberto Vázquez Sánchez

La Habana, 2 de diciembre 2022

“Año 64 de la Revolución”

Dedicatoria

Dedico el presente trabajo de diploma a mis padres por su amor, confianza y apoyo incondicional, a mis amigos por estar a mi lado en esto largo camino universitario y a mis hermanos que junto a mis padres son lo que más quiero en este mundo.

Agradecimientos

A mi madre y mi padre, son tantas cosas por lo que agradecerles que no las puedo resumir en unas oraciones, pero tengan claro, que todo lo que soy es gracias a ellos.

A mis amistades que me apoyaron en cada momento que lo necesité.

A mis tutores, que me han ayudado tanto en todo el proceso, gracias por brindarme su apoyo.

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis que tiene por título: Vista de detalle para el módulo de planos técnicos del sistema AsiXmec y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo. Para que así conste firmo la presente a los 2 días del mes de diciembre del año 2022.

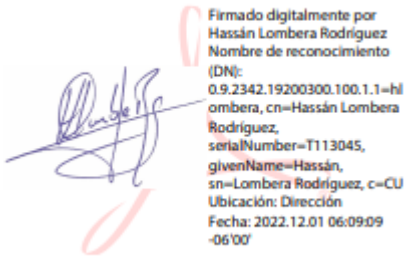
Arian Martínez Camejo

Firma del Autor

Ángel Alberto Vázquez Sánchez

Firma del Tutor

Hassán Lombera Rodríguez



Firma del Tutor

Datos de Contacto

Autor: Arian Martínez Camejo

Correo: arianmc@estudiamtes.uci.cu

Teléfono: 54550322

Tutor: M.Sc. Ángel Alberto Vázquez Sánchez

Correo: aavazquez@uci.cu

Tutor: Dr.C. Hassán Lombera Rodríguez

Correo: hlombera@uci.cu

Resumen

El presente trabajo tiene como objetivo desarrollar un módulo, con las tecnologías disponibles de código abierto, para brindarle al sistema AsiXmec la funcionalidad de una vista de detalle. En el mercado internacional existen muchos sistemas como lo son AutoCad, FreeCad, AutoDesk entre otros para el diseño asistido por computadora que poseen este módulo de vista de detalle; pero estas aplicaciones son de difícil acceso a causa del bloqueo impuesto contra Cuba y los altos precios de estas herramientas. Son innumerables los usos industriales de estas herramientas ya que con ellas se reduce el tiempo de creación de planos de detalles. La solución propuesta fue desarrollada con el lenguaje de programación C++, framework Qt y el IDE de desarrollo Qt Creator, y para guiar el proceso de desarrollo se empleó como metodología AUP en su versión UCI. Como resultado del proceso de investigación y desarrollo se obtuvo un módulo capaz de crear una vista de detalle a partir una vista base, dándole cumplimiento al objetivo planteado.

Palabras clave: *diseño asistido por computadora; planos de detalles; vista de detalle*

Abstract

The objective of this work is to develop a module, with the available open source technologies, to provide the AsiXmec system with the functionality of a detail view. In the international market there are many systems such as AutoCad, FreeCad, AutoDesk among others for computer aided design that have this detail view module; but these applications are difficult to access because of the blockade imposed against Cuba and the high prices of these tools. The industrial uses of these tools are innumerable since they reduce the time to create detail drawings. The proposed solution was developed with the C++ programming language, Qt framework and the Qt Creator development IDE, and to guide the development process, AUP in its UCI version was used as methodology. As a result of the research and development process, a module capable of creating a detail view from a base view was obtained, fulfilling the proposed objective.

Keywords: *computer-aided design; details plans; details views*

Índice

Índice de Figuras

Índice de Tablas

Introducción

El desarrollo y el uso de las Tecnologías de la Información y las Comunicaciones (TIC) está siendo uno de los principales motores de la economía del conocimiento, muy cercana a la economía de la información. Estas tecnologías les ofrece amplias oportunidades a las organizaciones, tales como facilitar la expansión del abanico de productos, la mejora del servicio al cliente, etc. Debido al aumento considerable de estas tecnologías, hoy en día se han desarrollado muchos sistemas en el área de las industrias para lograr una mayor eficiencia en el diseño de las piezas mecánicas, tales como lo son los sistemas de Diseño Asistido por Computadora o sistemas CAD [1] .

El software CAD se utiliza para aumentar la productividad del diseñador, mejorar la calidad del diseño, mejorar las comunicaciones a través de la documentación y crear una base de datos para la fabricación. También se puede considerar al CAD como una técnica de dibujo. Estas herramientas se pueden dividir básicamente en programas de modelado en 2D y 3D. Las herramientas de dibujo en 2D se basan en entidades geométricas vectoriales (puntos, líneas, arcos y polígonos) con las que se puede operar a través de una interfaz gráfica. Los modeladores en 3D añaden superficies y sólidos [1].

Un componente importante de los Sistemas Asistidos por Computadora (CAD) lo es el módulo de planos de fabricación, en el cual se especifican los detalles de las partes y piezas mecánicas[1]. Una de las funcionalidades más importantes de este tipo de módulo es la llamada vista de detalle, la cual le permite ver al diseñador a escalas de detalle más precisas una parte de la pieza que está diseñando. Debido a que la mayoría de los sistemas que implementan esta funcionalidad de vista de detalle en el módulo de planos técnicos son de difícil acceso ya que hay que pagar para poder obtenerlas y son de código cerrado como lo son AutoCAD, SolidWork y AutoDesk, en la Facultad 4 de la Universidad de las Ciencias Informáticas (UCI), se está desarrollando el proyecto de Investigación y Desarrollo “Plataforma para el modelado paramétrico en ingeniería basado en tecnologías de código abierto” AsiXmec. Este incorpora un modelador geométrico y paramétrico basados en el paradigma de diseño mediante características e historial de operaciones. AsiXmec está pensado para acortar el ciclo de diseño, en un área caracterizada por una elevada competitividad.

Debido a la situación antes descrita se tiene el siguiente problema investigativo:

El módulo de planos técnicos del sistema AsiXmec carece de la funcionalidad de vista de detalles.

Problema investigativo:

¿Cómo implementar una vista de detalles para el módulo de plano del sistema AsiXmec?

Objeto de Estudio:

Se centra en los módulos de planos técnicos en los sistemas CAD.

Objetivo General:

Desarrollar un módulo que provea al sistema AsiXmec de la funcionalidad vista de detalles, a partir de una vista base o proyectada de un modelo concreto.

Campo de Acción:

Está enmarcado en las vistas de detalles de los módulos de planos técnicos en los sistemas CAD.

Preguntas investigativas:

¿Qué se debe tener en cuenta para el desarrollo de la funcionalidad del módulo de planos del sistema AsiXmec?

¿Definiciones necesarias para el desarrollo de la funcionalidad del módulo?

¿Qué estrategia se debe realizar con el objetivo de tener la funcionalidad del módulo con todos los requisitos solicitados?

¿Cómo realizar las pruebas del software para el módulo?

Tareas de investigación

Análisis de sistemas CAD existentes en el ámbito nacional e internacional.

Selección de herramientas, tecnologías y metodología a utilizar en el desarrollo de la funcionalidad del módulo del sistema.

Implementación de la funcionalidad del módulo del sistema.

Validación del módulo a partir de la ejecución de las pruebas de software.

Métodos y Técnicas

Teóricos:

- **Método Histórico–Lógico:** Se utilizó para analizar a nivel nacional e internacional sistemas informáticos que tengan similitud con el que se va a implementar.
- **Método Analítico-Sintético:** Se utilizó para analizar las bibliografías disponibles acerca del tema a desarrollar.

Empíricos:

- **Consulta de información:** se utilizó para investigar todos los conceptos relacionados con el proyecto.
- **Entrevistas:** se utilizó para la obtención de información durante la investigación.

Capitulo I. Fundamentación Teórica

Introducción

En el actual capítulo se hace referencia de forma detallada los conceptos relacionados con el dominio del problema y se presenta las bases teóricas fundamentales relacionadas con las vistas de detalles. Se plasma un estudio a nivel nacional e internacional de los sistemas que presenten semejanza con el sistema propuesto, además se describen las tecnologías, herramientas, metodología y lenguajes que permiten dar solución al caso de estudio.

1.1 Conceptos asociados al tema:

Los conceptos asociados a un tema son aquellos que nos permiten tener una breve descripción o caracterización acerca de un tema que se quiere abordar. A continuación, se ofrecen algunos de los conceptos relacionados con el tema de investigación:

Dibujo técnico: es aquel que engloba representaciones de todo tipo de elementos mecánicos con el propósito de proporcionar información suficiente para facilitar su análisis [2].

Módulo de planos técnicos: es aquel que proporciona información acerca de una pieza que está en desarrollo para facilitar su análisis, ayudar a elaborar su diseño y posibilitar su futura construcción y mantenimiento [2].

Vista de detalles: Una vista de detalle es una vista proyectada y generada a partir de una vista de dibujo existente (Figura 1). En esta se muestra una parte específica de la vista de dibujo (desde la que se generó) en una escala superior [3].

Estos conceptos permitieron definir de forma precisa el enfoque de la investigación, ya que dieron respuesta a problemas de conocimiento relacionados con el tema a investigar.

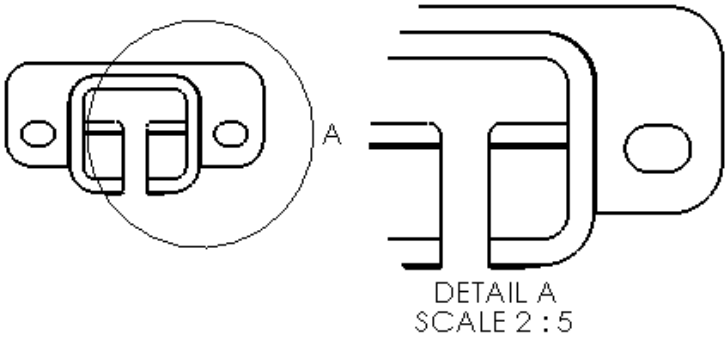


Figura 1 Ejemplo de vista de detalles. Fuente:[4]

1.2 Análisis de Sistemas similares

Como parte de la investigación se llevó a cabo un estudio de sistemas similares tanto nacional como internacional, considerando los sistemas informáticos que implementan vistas de detalles dentro de sus funcionalidades buscando así referencias y posibles soluciones a la problemática. Entre los sistemas encontrados se encuentra:

1.2.1 Nacionales:

Software Ingeniero: es un sistema que automatiza el proceso de diseño de piezas complejas como engranajes cilíndricos de dientes rectos, resortes helicoidales de tracción, extensión y torsión, árboles escalonados entre otros. Este fue creado por el grupo de investigación SIPII perteneciente al Departamento de Ciencias Básicas de la Facultad 4. Está basado en los sitemas asistidos por computadora (CAD), y para su creación se utilizó el lenguaje de programación C++ y las tecnologías Qt Framework, Open CASCADE y Qt Creator [5].

1.2.2 Internacionales:

AutoCAD: es un software de diseño asistido por computadora utilizado para dibujo 2D y modelado 3D. Actualmente es desarrollado y comercializado por la empresa Autodesk, reconocido a nivel internacional por sus amplias capacidades de edición, que hacen posible el dibujo digital de planos de edificios o la recreación de imágenes en 3D; es uno de los programas más usados por arquitectos, ingenieros, diseñadores industriales y otros para [6] :

- Dibujar y anotar geometría 2D y modelos 3D con sólidos, superficies y objetos de malla.

- Automatizar tareas como, por ejemplo, la comparación de dibujos, el recuento de objetos, la adición de bloques, la creación de tablas de planificación y mucho más.
- Crear un espacio de trabajo personalizado para maximizar la productividad con aplicaciones complementarias y API.

Además de acceder a comandos desde la solicitud de comando y las interfaces de menús, AutoCAD proporciona interfaces de programación de aplicaciones (API) que se pueden utilizar para determinar los dibujos y las bases de datos. Este dentro de sus funcionalidades tiene la opción de realizar vistas de detalles la cual dispone de múltiples opciones como: seleccionar si la vista será circular o rectangular, el tipo de borde de la vista el cual puede ser liso o dentado; la vista se crea con un identificador por defecto, pero este puede ser editado posteriormente y elegir si es visualizado o no en la vista, su color también puede ser cambiado y se puede ajustar su posición dentro del plano [6].

Autodesk Inventor: ofrece un conjunto de herramientas profesionales ideales para el diseño mecánico, simulación, visualización y documentación de productos en 3D, permite la integración de datos en 2D y 3D en un único entorno, creando una representación virtual del producto final que le permitirá inspeccionar la forma, el ajuste y el funcionamiento del producto en cualquier momento del proceso de diseño antes de su fabricación. El software ha sido diseñado para facilitar los procesos de trabajo de un equipo, optimizar sinergias y flujos de trabajo. Este contiene una combinación potente de capacidades de diseño paramétrico, directo, de formas libres y basado en reglas, el cual una gran cantidad de funciones como diseño paramétrico de piezas y ensamblajes, simulación, visualización, automatización, bibliotecas de elementos normalizados y bocetaje, que en otros programas CAD son secundarias [7].

Autodesk Inventor dispone de una funcionalidad para realizar vistas de detalle la cual brinda múltiples opciones como: ajustar la escala de la vista de detalle inicial calculando el espacio disponible en el plano de detalle donde se está trabajando; permite visualizar la totalidad de la vista aun si la parte del área seleccionada no forma parte del objeto sobre el cual se desea realiza la vista; permite la creación de líneas de conexión que enlazan el área seleccionada para la vista con ampliación, si selecciona el tipo de línea de corte Suavizado, se puede optar por visualizar un contorno plano (circular o rectangular) alrededor de la vista de detalle resultante. Los tres objetos de anotación (el borde, el contorno y la línea de conexión) forman un objeto de anotación.

El aspecto por defecto del contorno de la vista de detalle, las flechas, el estilo de texto del identificador y la orientación del texto de la directriz se especifica en el estilo Anotación de vista [7].

SOLIDWORKS: es un software de diseño CAD 3D para modelar piezas y ensamblajes en 3D y planos 2D, el cual ofrece un abanico de soluciones para cubrir los aspectos implicados en el proceso de desarrollo del producto, como lo son: crear, diseñar, simular, fabricar, publicar y gestionar los datos del proceso de diseño. Tiene la gran ventaja de ser un software de diseño paramétrico modular, más fácil de usar que su competencia ya que permiten visualizar el diseño en tiempo real, esta herramienta te presenta una visión de la forma en que se verá el diseño sin que realmente se realice. Puede ver cada parte de forma individual del diseño, ver propiedades de masa precisas y verificar la interferencia, lo que significa que no tendrá que construir / fabricar el producto antes de ver cualquier error, ahorrando tiempo y dinero y reduciendo la cantidad de prototipos necesarios. Contiene herramientas analíticas integradas y automatización de diseño para ayudar a estimular el comportamiento físico como cinemática, dinámica, estrés, deflexión, vibración, temperaturas o flujo de fluidos para adaptarse a todo tipo de diseño. Permite además evaluar ensambles con un gran volumen de piezas, evaluar el impacto ambiental del diseño, simular virtualmente las condiciones y analizar el diseño en situaciones reales, además de optimizar su desempeño y generar documentación técnica como la planimetría de fabricación para el taller [8].

SOLIDWORKS dispone de una funcionalidad para crear una vista de detalle en un dibujo para visualizar una porción de una vista, normalmente a una escala mayor. Este detalle puede ser de una vista ortográfica, una no plana (isométrica), una vista de sección, una vista recortada, una vista explosionada de ensamblaje u otra vista de detalle. La función de crear vistas de detalle dispone de múltiples opciones como: cambio de escala, cambio del texto identificador de la vista, cambio de la fuente del identificador, inmovilización de la vista, creación de la vista con contornos irregulares, creación de líneas de conexión, seleccionar el estilo de la vista (circular o rectangular)[8].

Tabla 1.1 Comparación entre los sitemas encontrados

Sistemas	Objetivo	Vista de detalle	Planos	Mercado	Código Fuente	Plataforma
AutoCad	dibujo digital de planos de edificios	Si	2D/3D	Privativo	Cerrado	Windows
SolidWord	modelar piezas y ensamblajes	Si	2D/3D	Privativo	Cerrado	Windows/Linux
AutoDesk	diseño mecánico, simulación, visualización y documentación de productos	Si	2D/3D	Privativo	Cerrado	Windows/Linux
Ingeniero	Diseño de piezas complejas	no	2D	Libre	Abierto	Windows/Linux

1.2.3 Conclusiones Parciales

El estudio realizado a las herramientas antes mencionadas evidencio la necesidad de desarrollar un módulo de vista de detalle para el sistema AsixMec, debido a que son sistemas privativos, de código cerrado y al bloqueo impuesto a Cuba este no puede utilizarlas ya que no puede obtener su licencia. Además, el desarrollo de esta funcionalidad le ofrece soberanía tecnológica a nuestro país.

1.3 Lenguaje de Programación

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Se asume generalmente que la traducción de las instrucciones a un código que comprende la computadora debe ser completamente sistemática. Estos se componen de un conjunto de reglas sintácticas y semánticas que permiten expresar

instrucciones que luego serán interpretadas. Para llevar a cabo la implementación se decide escoger como lenguaje de programación C++ ya que el sistema a donde se va integrar el módulo está desarrollado en este lenguaje, lo que va a permitir una correcta integración de este.

Las principales características del lenguaje C++ son [9]:

- Compatibilidad con bibliotecas**:** A través de bibliotecas hay muchas funciones que están disponible y que ayudan a escribir código rápidamente.
- Orientado a Objetos**:** El foco de la programación está en los objetos y la manipulación y configuración de sus distintos parámetros o propiedades.
- Rapidez**:** La compilación y ejecución de un programa en C++ es mucho más rápida que en la mayoría de lenguajes de programación.
- Compilación**:** En C++ es necesario compilar el código de bajo nivel antes de ejecutarse, algo que no ocurre en otros lenguajes.
- Punteros**:** Los punteros del lenguaje C, también están disponibles en C++.
- Didáctico**:** Aprendiendo programación en C++ luego es mucho más fácil aprender lenguajes como Java, C#, PHP, Javascript, etc.

Las principales ventajas de programar en C++ [9] son:

- Alto rendimiento: Es una de sus principales características, el alto rendimiento que ofrece. Esto es debido a que puede hacer llamadas directas al sistema operativo, es un lenguaje compilado para cada plataforma, posee gran variedad de parámetros de optimización y se integra de forma directa con el lenguaje ensamblador.
- Lenguaje actualizado: A pesar de que ya tiene muchos años, el lenguaje se ha ido actualizando, permitiendo crear, relacionar y operar con datos complejos y ha implementado múltiples patrones de diseño.
- Multiplataforma: Se encuentra disponible tanto para dispositivos y plataformas iOS y Android, además de en Windows, Windows Store y Xbox.
- Extendido: C y C++ están muy extendidos. Casi cualquier programa o sistema están escritos o tienen alguna parte escrita en estos lenguajes (desde un navegador web hasta el propio sistema operativo).

Las principales desventajas de C++ es que se trata de un lenguaje muy amplio (con muchos años y muchas líneas de código), tiene que tener una compilación por plataforma y su depuración se complica debido a los errores que surgen. Además, el manejo de librerías es más complicado que otros lenguajes como Java o .Net y su curva de aprendizaje muy alta[9].

1.4 Herramientas y Tecnologías

Para el desarrollo de la propuesta solución se tendrán en cuenta un conjunto de herramientas y tecnologías para agilizar este proceso. Para esto, se escogieron aquellas que garanticen la correcta integración del módulo a desarrollar al sistema AsiXmec.

1.4.1 Open Cascade Aplication Framework: Open CASCADE Technology (OCCT) es una biblioteca de clases de C++ orientada a objetos diseñada para la producción rápida de aplicaciones CAD/CAM/CAE específicas de dominio. Es una aplicación típica desarrollada con OCCT que se ocupa del modelado geométrico bidimensional o tridimensional (2D o 3D) en sistemas de diseño asistido por computadora (CAD) especializados o de propósito general, aplicaciones de fabricación o análisis, aplicaciones de simulación o incluso herramientas de ilustración [10].

Open CASCADE Aplication Framework(OCAF) es un Framework de Desarrollo Rápido (RAD) utilizado para especificar y organizar los datos de la aplicación, para esto OCAF provee [10]:

- Datos comunes listos para usar compatibles con la mayoría de las aplicaciones CAD/CAM
- Un protocolo de extensión escalable para implementar nuevos datos específicos de la aplicación
- Una infraestructura:

  Para adjuntar cualquier dato a cualquier elemento topológico

  Vincular datos producidos por diferentes aplicaciones

  Para registrar el proceso de modelado, historial de creación o parámetros utilizados para realizarlas modificaciones.

También diseñada para ser verdaderamente modular y extensible, proporcionando clases de C++ para [10]**:**

- Estructuras de datos básicos (modelado geométrico, visualización, selección interactiva y servicios específicos de aplicaciones);
- Algoritmos de modelado;
- Trabajar con datos de malla (facetados);
- Interoperabilidad de datos con formatos neutros (IGES, STEP)

1.4.2 QT Framework: es un software multiplataforma para crear interfaces gráficas de usuario, así como aplicaciones multiplataforma que se ejecutan en varias plataformas de software y hardware como Linux, Windows, macOS, Android o sistemas integrados con poco o ningún cambio en la base de código, también es compatible con varios compiladores, incluido el compilador GCC C++, la suite Visual Studio, PHP a través de una extensión para PHP5y tiene un amplio soporte de internacionalización [11].

Qt está escrito en lenguaje C++, posee un preprocesador, MOC (Meta-Object Compiler) que se utiliza para extender el lenguaje C++ con funciones como Signals y Slots. Antes del paso de compilación, MOC analiza los archivos de origen escritos en Qt-extended C++ y genera código fuente C++ estándar compatibles de ellos. Por lo tanto, el propio framework y las aplicaciones/librerías que lo utilizan pueden ser compilados por cualquier compilador compatible con C++ estándar como Clang, GCC, ICC, MinGW y MSVC [11].

1.5.3 QT Creator: es un entorno de desarrollo integrado (IDE) multiplataforma C++, JavaScript y QML que simplifica el desarrollo de aplicaciones GUI. Es parte del SDK para el marco de desarrollo de aplicaciones de la GUI de Qt y utiliza la API de Qt, que encapsula las llamadas a funciones de la GUI del SO host. Incluye un depurador visual y un diseñador de formularios y diseño de GUI WYSIWYG integrado. El editor tiene funciones como resaltado de sintaxis y autocompletado. Qt Creator usa el compilador C++ de GNU Compiler Collection en Linux, en Windows, puede usar MinGW o MSVC con la instalación predeterminada y también puede usar Microsoft Console Debugger cuando se compila desde el código fuente. La configuración de compilación permite al usuario cambiar entre objetivos de compilación, diferentes versiones de Qt y configuraciones de compilación. Para objetivos de dispositivos móviles, Qt Creator

puede generar un paquete de instalación, instalarlo en un dispositivo móvil que esté conectado a la computadora de desarrollo y ejecutarlo allí [12].

Qt Creator integra las siguientes herramientas además del editor de código [12]:

- **qmake**: es el sistema de compilación estándar de Qt y, como tal, se encuentra integrado en Qt Creator y puede seleccionarse desde el menú. No obstante, el IDE es compatible también con otros sistemas. Para utilizar Qbs, solo tendrás que abrir un archivo.
- **Qt Designer**: Qt Designer es el programa nativo para diseñar y desarrollar interfaces gráficas de usuario con ayuda de Qt Widgets. El editor visual permite colocar y ajustar los widgets como se desee.
- **Qt Linguist**: las aplicaciones pueden localizarse directamente en Qt Creator. Desarrolladores, traductores y gestores de entregas encontrarán aquí las herramientas apropiadas.
- **Qt Assistant**: con Qt Assistant, Qt Creator brinda un acceso rápido a documentación oficial del framework. Para ello, se dispone de una función de ayuda.

Se escogieron estas herramientas para el desarrollo de la solución porque son parte de las herramientas de desarrollo que se utilizaron para desarrollar el proyecto AsiXmec y así garantizar una plena compatibilidad de la nueva funcionalidad.

1.5.4 Doxygen: es una herramienta para generar documentación a partir de códigos C++ anotados, pero también es compatible con otros lenguajes de programación populares como C, PHP, Java, Python. Permite extraer la estructura del código fuente no documentados. Doxygen también puede visualizar las relaciones entre los distintos elementos mediante la inclusión de gráficos de dependencia, diagramas de herencia y diagramas de colaboración, que se generan todos automáticamente; está desarrollado para Mac OS X y Linux, pero está configurado para ser altamente portátil. Como resultado, también se ejecuta en la mayoría de los otros tipos de Unix. Además, hay disponibles ejecutables para Windows [13].

Pasos para crear una correcta documentación usando esta herramienta [13]:

- Comentar cada declaración de clase, método, función, objeto o variable en el código usando las palabras claves que trae esta herramienta ejemplo: @file nombre del archivo contenedor del módulo.

- Abrir el Doxygen y cargar el proyecto en este.
- Configurar cada uno de los parámetros que trae esta herramienta (Project, Mode, Output y Diagrams).
- Una vez configurado cada uno de los parámetros se le da Run a la herramienta y esta muestra la documentación en dependencia de la configuración que haya hecho el usuario.

Se seleccionó la herramienta Doxygen para mantener una correcta documentación del código y así garantizar limpieza y organización en el código; además facilita la comprensión de este para futuros desarrolladores.

1.6 Metodología AUP-UCI

La metodología seleccionada para el desarrollo de la investigación fue AUP-UCI porque es apropiada para proyectos pequeños, el negocio está bien definido por parte del usuario y se adapta perfectamente a los objetivos de la investigación.

AUP-UCI se divide en tres fases fundamentales [14]:

Inicio: 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.

Ejecución (Elaboración, Construcción, Transición): En esta fase 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.

Cierre: En esta fase se analizan tanto los resultados del proyecto como la ejecución y se realizan las actividades formales de cierre del proyecto.

La metodología de software AUP-UCI a partir de que el modelado de negocio propone tres variantes a utilizar en los proyectos, como son: CUN (Casos de uso del negocio), DPN (Descripción de proceso de negocio) o MC (Modelo conceptual) y existen tres formas de encapsular los requisitos los cuales son: CUS (Casos de uso del sistema), HU (Historias de usuario), DRP (Descripción de requisitos por proceso), surgen cuatro escenarios para modelar el sistema en los proyectos, los cuales son [14]:

- Escenario No 1: Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.
- Escenario No 2: Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.
- Escenario No 3: Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.
- Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con HU.

A partir del análisis e investigación realizada, se utilizará el escenario #4 de la metodología AUP-UCI, debido a que es apropiada para este tipo de proyectos pequeños permitiendo disminuir las probabilidades de fracaso, por ser un producto de fácil uso.

1.7 AsiXmec:

Es un modelador geométrico y paramétrico basado en Feature e historial de operaciones que permite a los usuarios manipular, crear, editar, importar y exportar archivos STEP, IGES, STL, compatibles con otras aplicaciones para Linux como DraftSight, LibreCAD y para Windows como AutoCAD, SolidWorks e Inventor, entre otras de las más conocidas. AsiXmec se ha desarrollado con lenguaje C++, basado en la plataforma DISEM para el desarrollo de la línea CAD-Computer-aided engineering (CAE), utiliza el framework Open CASCADE, marco de desarrollo Qt-5 y soporte para plugins. Cuenta con un solver de restricciones que utiliza la biblioteca Eigen, permite el diseño asistido por snap e identificación automática de caras. Su interface gráfica tiene soporte multilenguaje (inglés y español)[15]. Con una arquitectura basada en plugins y soporte multilenguaje, posee una elevada capacidad de extensibilidad y adaptabilidad para la incorporación de nuevos módulos y la creación de sistemas especializados o a la medida. La diversidad de los modelos que se pueden obtener posibilita su uso en una amplia variedad de industrias. El software ha sido desarrollado con bibliotecas multiplataforma, aunque hasta ahora solo ha sido compilado en Ubuntu, pero evitando cualquier dependencia que en el futuro que pueda frenar la compilación en Windows. El objetivo con el que se desarrolló este software es proveer al usuario de una herramienta libre de coste capaz de crear, modificar y visualizar modelos en 2D y 3D [15]. A pesar de poseer todas estas funcionalidades AsiXmec carece de otras que son necesarias para aumentar su alcance y competitividad, ejemplo de esto es la ausencia

de un módulo de planos técnicos donde el usuario pueda realizar vistas de detalle, vistas explosionadas o anotaciones en dichas vistas.

Conclusiones Parciales

Los sistemas CAD son una herramienta imprescindible para todo tipo de industrias especialmente para las que tienen un alto contenido de fabricación de partes y piezas. Con este fin los sistemas CAD poseen un módulo de planos encargado de crear archivos de dibujo donde queden reflejados los detalles de manufactura de dichas partes y piezas. Después del estudio del estado del arte y la presencia de la funcionalidad de creación de vistas de detalle en sistemas CAD populares se decidió que la propuesta de solución debía tener elementos como: creación de una etiqueta identificativa para las vistas de detalle, seleccionar el espacio de proyección de las vistas, modificar la escala de la vista, seleccionar la forma de recortado de estas entre otros.

Además, la metodología a utilizar es la AUP-UCI ya que se adapta perfectamente a los objetivos de la investigación que es añadir funcionalidades a un módulo de un sistema ya existente. Dentro de las herramientas elegidas para el desarrollo de la funcionalidad se encuentran Open Cascade Aplication Framework, Qt Creator, QT Framework y Doxygen ya que son las más utilizadas en este tipo de proyecto y nos permiten el modelado geométrico bidimensional o tridimensional (2D o 3D) en sistemas de diseño asistido por computadora (CAD) especializados o de propósito general.

Capitulo II. Propuesta de Solución

Introducción

En este capítulo se presentar una noción propia del problema, así como la propuesta para resolverlo definiendo los requisitos funcionales y no funcionales de la solución, realizando una descripción de la arquitectura del sistema AsiXmec y se expone la propuesta arquitectónica para incluir la solución. También se incluyen los artefactos ingenieriles generados en el diseño de la solución.

2.1 Ingeniería de Requisitos

La Ingeniería de requisitos es muy utilizada en los procesos de desarrollo de software, ya que a partir de estas se obtienen los requisitos necesarios para el desarrollo de un sistema. Una de las técnicas de obtención de requisitos que tiene esta disciplina es la técnica de educción de requisitos. La educción de requisitos es un estudio profundo de una necesidad tecnológica que tiene una empresa, organización o negocio. En este proceso, se realiza un análisis exhaustivo del sistema que se va a desarrollar. Se definen y aplican técnicas que permitan analizar los requisitos necesarios para su buen desarrollo. De esta forma, se logra reconocer y entender cuáles son las verdaderas necesidades que el sistema debe solucionar [16].

2.1.1 Objetivo:

Conocer los requisitos relevantes para que el sistema sea lo más completo posible y lograr un consenso entre los implicados sobre estos requisitos y comprender los deseos y necesidades de los implicados.

2.1.2 Fuentes de Requisitos

La forma de adquirir los requisitos de un sistema utilizando esta técnica son [16] :

- **Implicados:** Un implicado es definido como una persona u organización que tiene una influencia directa o indirecta sobre los requisitos de un sistema. La influencia indirecta incluye además situaciones donde una persona u organización se ve afectada por el sistema.
- **Documentación:** Se usan para transferir conceptos entre humanos a lo largo del tiempo y la distancia.

- **Sistemas existentes:** En el contexto (tanto directo como amplio) de un sistema, se pueden identificar otros sistemas como fuentes de requisitos.

2.1.3 Técnicas para la educción de requisitos

Las técnicas para la educción de requisitos son [16]:

- Técnicas tradicionales: Cuestionarios, Surveys, Entrevistas (de comienzo y final abierto, estructuradas) y Análisis de documentos (formularios, organigramas, modelos, estándares, manuales, normas, etc).
- Técnicas de elicitación grupales**:** Brainstorming, Focus groups y RAD/JAD.
- Prototipos: Solo o combinado con otras técnicas.
- Técnicas orientadas por modelos: Modelos basados en objetivos y modelos basados en escenarios.
- Técnicas cognitivas**:** Laddering, Card Sorting y Repertory Grids.
- Técnicas contextuales**:** Métodos etnográficos (observación del participante), Etnometodología y Análisis de conversación (estudio de conversación e interacción.

Al tomar como fuente de requisitos los sistemas existentes como lo son AutoCad, Autodesk Inventor y Solidworks que implementan las vistas de dalles como una de sus funcionalidades y son sistemas que ofrecen un conjunto de herramientas profesionales ideales para el diseño mecánico, simulación, visualización y documentación de productos en 2D y 3D; y realizado las técnicas de educción de requisitos (Análisis de documentos) y el prototipo, se llegó a que los requisitos funcionales a implementar en la solución del sistema son:

- RF1: Crear vista de detalle
- RF2: Configurar escala
- RF3: Modificar escala
- RF4: Crear etiqueta identificativa
- RF5: Modificar etiqueta identificativa
- RF6: Seleccionar espacio de proyección
- RF7: Seleccionar la forma de recortar
- RF8: Modificar la forma de recortar
- RF9: Mostrar espacio de proyección

2.2 Historias de Usuario

Las Historias de Usuario son una explicación general e informal de las funcionalidades de un software escritos desde la perspectiva del usuario final. Su propósito es articular cómo proporcionará una función de software valor al cliente. Se decidió agrupar los requisitos funcionales en historias de usuario porque la solución a desarrollar no es muy extensa y el negocio está bien definido por parte del cliente.

Tabla 2.1 Historia de Usuario # 1

Historia de Usuario	
Número:1	Nombre: Crear vista de detalles
Usuario: Usuario	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Media
Puntos estimados: 1	Iteración Asignada: 1
Programador Responsable: Arian Martínez Camejo	
Descripción: Después que el usuario haya seleccionado todos los campos necesarios para la creación de la vista de detalle y presione el botón aceptar se crea la vista en el plano con todas las características previamente seleccionadas(escala, etiqueta identificativa, forma de recortado y espacio de proyección).	
Observaciones: La vista se ubica en el espacio elegido por el usuario.	

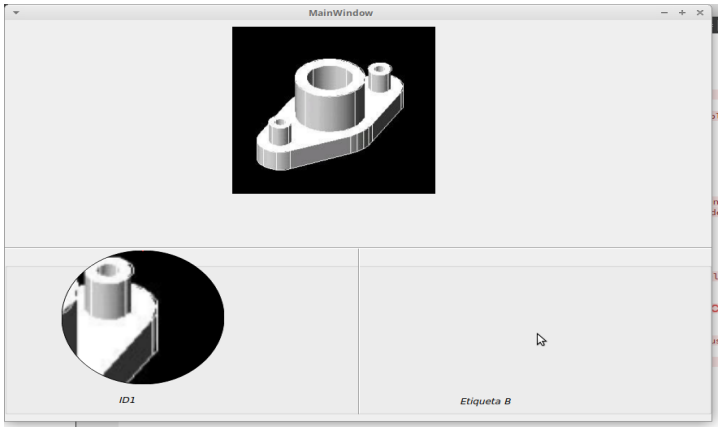


Figura 2 Prototipo de Interfaz para la historia de usuario # 1

Tabla 2.2 Historia de Usuario # 2

Historia de Usuario	
Número:2	Nombre: Configurar escala
Usuario: Usuario	
Prioridad en Negocio: Media	Riesgo en Desarrollo: Media
Puntos estimados: 0.7	Iteración Asignada: 1
Programador Responsable: Arian Martínez Camejo	
Descripción: Permite al usuario seleccionar de entre varias escalas(1:2, 1:5, 2:1 y 5:1) para visualizar la vista.	
Observaciones: El valor por defecto de la escala de la vista será 1:2.	

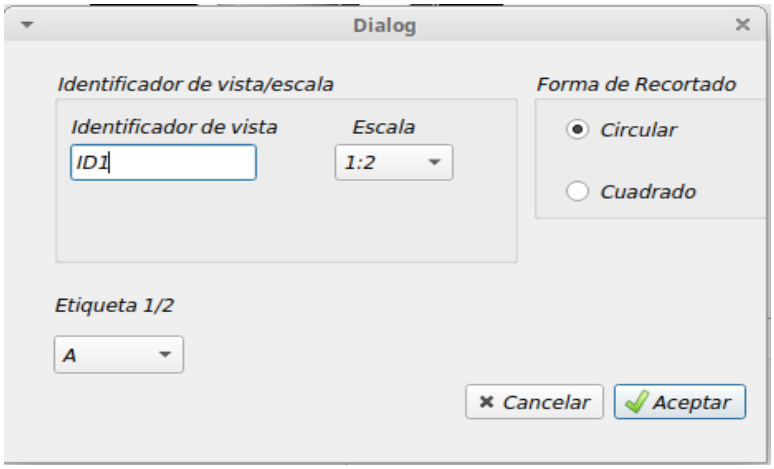


Figura 3 Prototipo de interfaz para las historias de usuario # 2 y # 3

Tabla 2.3 Historia de Usuario # 3

Historia de Usuario	
Número: 3	Nombre: Modificar escala
Usuario: Usuario	
Prioridad en Negocio: Baja	Riesgo en Desarrollo: Baja
Puntos estimados: 0.6	Iteración Asignada: 3
Programador Responsable: Arian Martínez Camejo	

Descripción: Permite al usuario cambiar la escala de una vista de detalle seleccionada.
Observaciones:

Tabla 2.4 Historia de Usuario # 4

Historia de Usuario	
Número:4	Nombre: Crear etiqueta identificativa
Usuario: Usuario	
Prioridad en Negocio: Baja	Riesgo en Desarrollo: Baja
Puntos estimados: 0.2	Iteración Asignada: 1
Programador Responsable: Arian Martínez Camejo	
Descripción: Permite al usuario crear una etiqueta para identificar la vista una vez creada. En caso de no escribir esta se creara con un identificador por defecto el cual será ID1.	
Observaciones:	

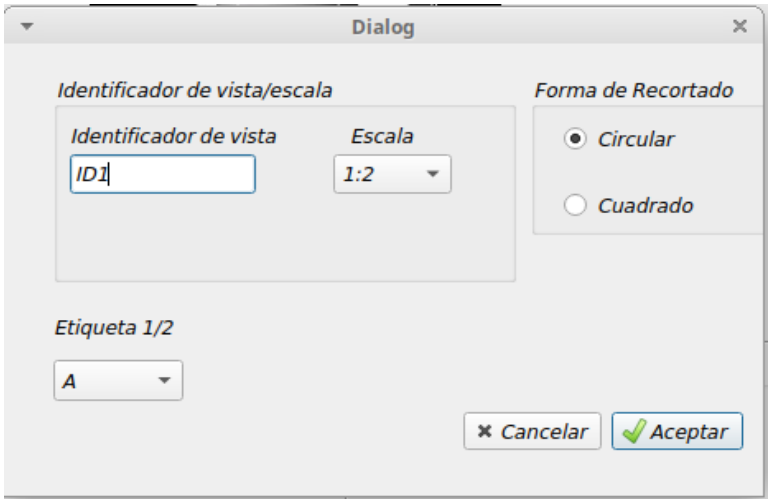


Figura 4 Prototipo de interfaz para las historias de usuario # 4 y # 5

Tabla 2.5 Historia de Usuario # 5

Historia de Usuario	
Número: 5	Nombre: Modificar etiqueta identificativa
Usuario: Usuario	
Prioridad en Negocio: Baja	Riesgo en Desarrollo: Baja

Puntos estimados: 0.2	Iteración Asignada: 2
Programador Responsable: Arian Martínez Camejo	
Descripción: Permite al usuario cambiar la etiqueta identificativa de una vista de detalle seleccionada.	
Observaciones:	

Tabla 2.6 Historia de Usuario # 6

Historia de Usuario	
Número: 6	Nombre: Seleccionar espacio de proyección
Usuario: Usuario	
Prioridad en Negocio: Baja	Riesgo en Desarrollo: Baja
Puntos estimados: 0.3	Iteración Asignada: 3
Programador Responsable: Arian Martínez Camejo	
Descripción: Permite al usuario seleccionar el espacio donde será proyectada la vista.	
Observaciones: El espacio estará dividido en 2 partes (Etiqueta A , Etiqueta B).	

Tabla 2.7 Historia de Usuario # 7

Historia de Usuario	
Número: 7	Nombre: Seleccionar la forma de recortar
Usuario: Usuario	
Prioridad en Negocio: Baja	Riesgo en Desarrollo: Baja
Puntos estimados: 0.3	Iteración Asignada: 3
Programador Responsable: Arian Martínez Camejo	
Descripción: Permite al usuario seleccionar la forma en que se va a mostrar la vista de detalle seleccionada, esta puede ser circular o cuadrada.	
Observaciones: La forma por defecto será circular.	

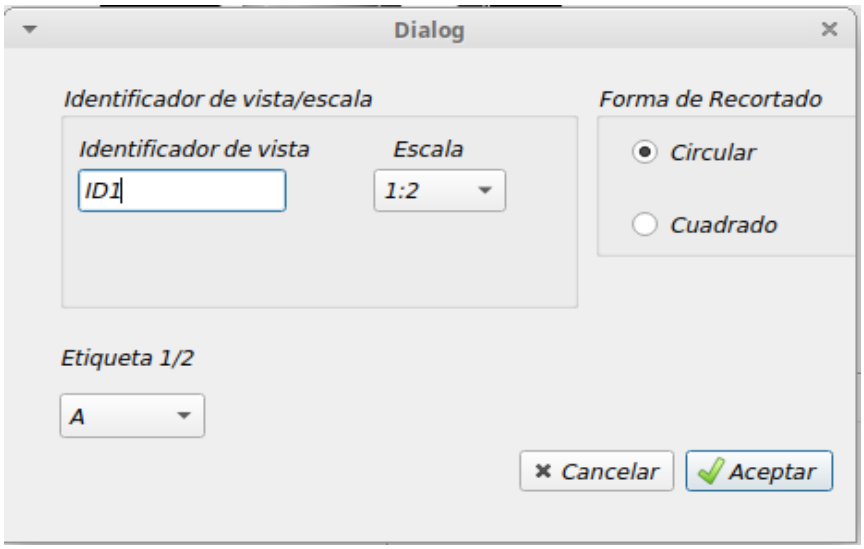


Figura 5 Prototipo de interfaz para las historias de usuario # 6, # 7 y # 8

Tabla 2.8 Historia de Usuario # 8

Historia de Usuario	
Número: 8	Nombre: Modificar la forma de recortar
Usuario: Usuario	
Prioridad en Negocio: Baja	Riesgo en Desarrollo: Baja
Puntos estimados: 0.3	Iteración Asignada: 2
Programador Responsable: Arian Martínez Camejo	
Descripción: Permite al usuario modificar la forma en que se vera la vista, esta puede ser circular o cuadrado.	
Observaciones	

Tabla 2.9 Historia de Usuario # 9

Historia de Usuario	
Número: 9	Nombre: Mostrar espacio de proyección
Usuario: Usuario	
Prioridad en Negocio: Baja	Riesgo en Desarrollo: Baja
Puntos estimados: 0.3	Iteración Asignada: 2
Programador Responsable: Arian Martínez Camejo	

Descripción: Permite al usuario visualizar los espacios donde serán proyectadas las vistas.
Observaciones

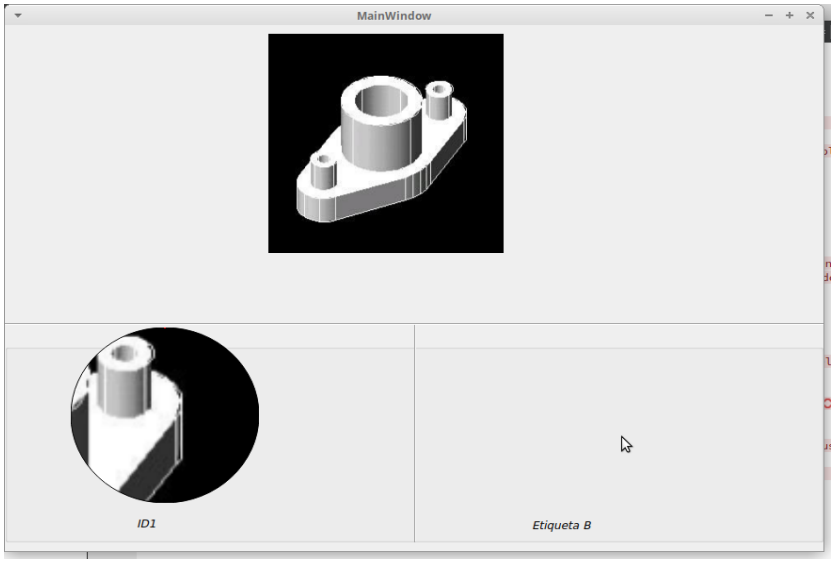


Figura 6 Prototipo de interfaz para la historia de usuario # 9

2.3 Requisitos No Funcionales

Los requisitos no funcionales explican los aspectos de calidad del sistema que se va a construir, estos pueden ser de tipo: rendimiento, portabilidad, usabilidad, etc. Los requisitos no funcionales, a diferencia de los requisitos funcionales, se implementan de forma incremental en cualquier sistema. Estos se obtuvieron a partir del estudio de sistemas similares y las características del propio sistema AsiXmec.

Software

- RNF1: El sistema operativo sobre el que se corra el software debe tener arquitectura de 64 bits.

Usabilidad

- RNF2: El software debe tener una interfaz intuitiva.

Hardware

- RNF3: El ordenador donde se corra el software debe tener un mínimo de 4GB de memoria RAM.
- RNF4: El ordenador donde se corra el software debe tener un microprocesador Intel core i5 o superior.

Portabilidad

- RNF5: El software solo funcionará sobre un sistema operativo Ubuntu 14.04 o superior o en cualquier distribución basada en este.

2.4 Propuesta arquitectónica

AsiXmec presenta una arquitectura basada en plugins (Figura 7) y se compone de dos módulos principales:

- **Módulo para el modelado en 2D:** Posee herramientas para el trabajo con las dimensiones de las entidades como son: definición de longitud, relaciones de ángulos, definición del diámetro de circunferencia, definición de radio de arco, definición de distancia entre entidades o modificar el valor de una dimensión. Permite la generación de copias de entidades 2D anteriormente creadas de forma libre o utilizando patrones como rectangular, circular o espejo.
- **Módulo de planos técnicos:** Es el encargado de construir planos de fabricación para partes y piezas creados en AsiXmec. El módulo posee la facilidad de generar archivos de dibujo en un número corto de pasos. Incluye la generación de vistas base vistas proyectadas vistas de sección vistas explosionadas vistas auxiliares vistas superpuestas cotas anotaciones símbolos y tablas.

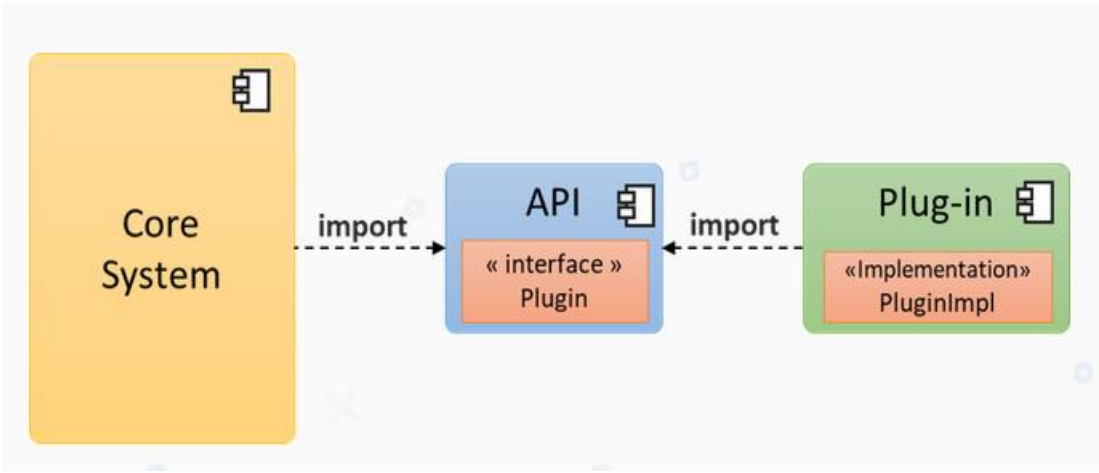


Figura 7 Arquitectura basada en plugins. Fuente: [17]

La propuesta de solución consiste en diseñar la solución de la funcionalidad para la generación de vistas de detalle del módulo de planos técnicos ya que actualmente este carece de esta.

2.5 Propuesta de diseño

Como propuesta de diseño de la solución se decidió utilizar el patrón Builder que es patrón de diseño creacional que permite construir objetos complejos paso a paso. Este patrón nos permite producir distintos tipos y representaciones de un objeto empleando

el mismo código de construcción (Figura 8). Para la realización de esta propuesta se tuvieron en cuenta aspectos como, el lenguaje de programación, las herramientas a utilizar y el entorno de desarrollo para obtener una integración exitosa del módulo a desarrollar al sistema AsiXmec.

El Builder Pattern es un tipo de patrón de diseño que sirve para resolver tareas de programación en una programación orientada a objetos. Los patrones Builder (o Constructor) facilitan a los desarrolladores el proceso de programación porque no han de rediseñar cada paso que se repite como una rutina de programa, en vez de rediseñar cada paso, pueden utilizar una solución establecida. Este patrón mejora tanto la seguridad de construcción como la legibilidad del código del programa, el objetivo del patrón es crear un objeto sin constructores conocidos, pero con una clase auxiliar [18].

Un patrón de diseño Builder distingue entre cuatro actores:

- **Director:** este actor construye el objeto complejo con la interfaz del constructor. Es consciente de los requisitos de secuencia del constructor. Con el director, se desvincula la construcción del objeto del cliente.
- **Builder:** ofrece una interfaz para crear los componentes de un objeto (o producto) complejo.
- **Specific builder:** crea las partes del objeto complejo, define (y gestiona) la representación del object, y mantiene la interfaz de salida del objeto.
- **Producto:** es el resultado de la “actividad” del Builder Pattern, es decir, el objeto que se construye.

El director supervisa el proceso decisivo del patrón Builder: la separación de la creación de un objeto/producto del cliente.

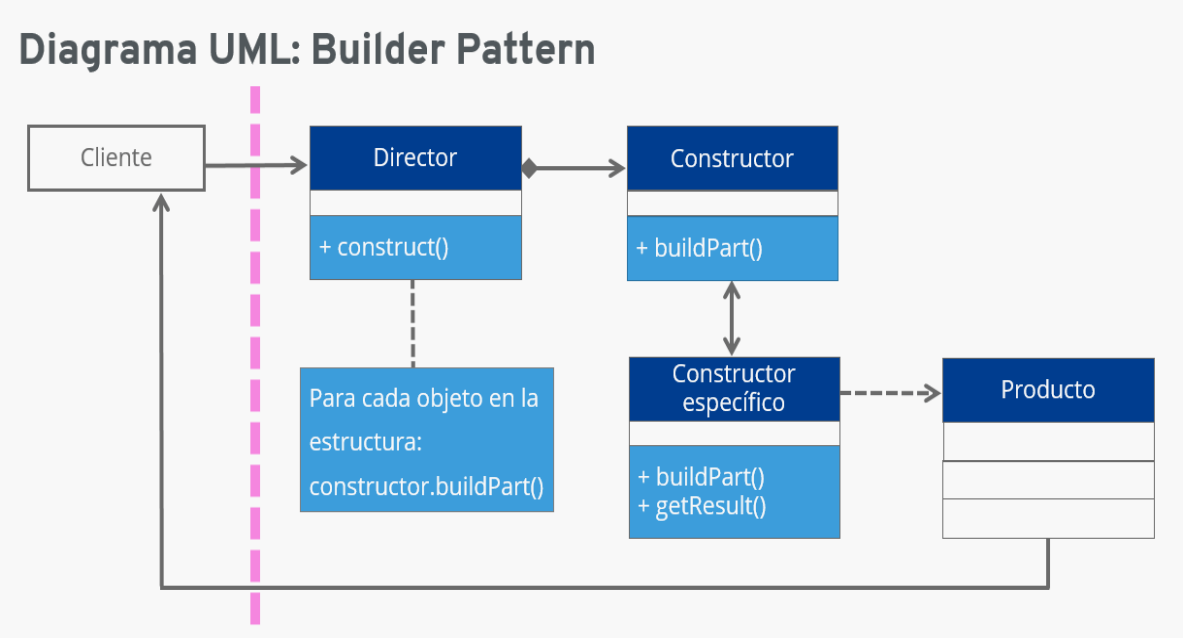


Figura 8 Estructura básica del patrón de diseño Builder en UML que ilustra las diversas relaciones que admite. Fuente: [19]

2.5.1 Ventajas del patrón de diseño Builder

La construcción y la representación se incorporan por separado. Las representaciones internas del constructor están ocultas para el director. Las nuevas representaciones como tal pueden integrarse fácilmente utilizando clases de constructores concretos. El proceso de construcción lo controla explícitamente el director. Si hay que hacer cambios, pueden hacerse sin consultar al cliente [18].

El siguiente diagrama de clases del diseño está compuesto por los siguientes elementos:

- La interfaz DetailViewBuilder declara los pasos de construcción necesarios para todas las vistas de detalle.
- La clase ConcreteDetailView es la responsable de proveer la implementación de los distintos pasos de construcción para las distintas vistas de detalle.
- La clase DetailView representa el resultado del proceso de construcción de la vista de detalle.
- La clase Director es la encargada de definir el orden en que se ejecutan los pasos de construcción.

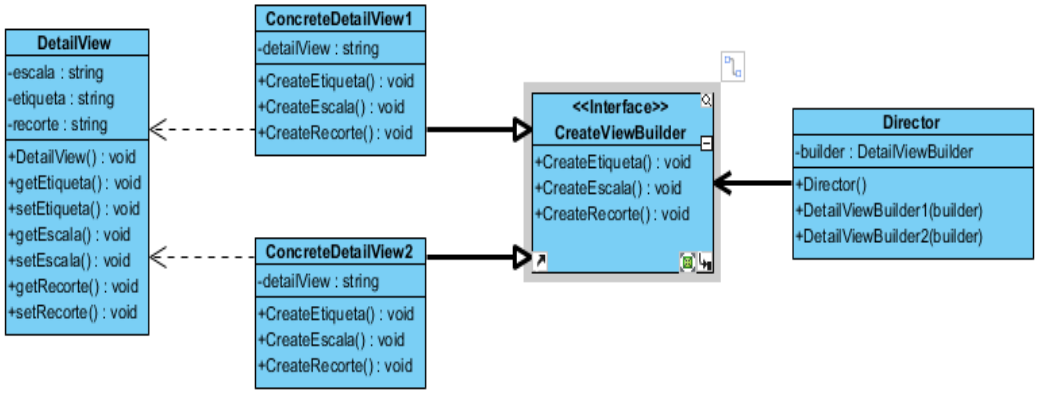


Figura 9 Diagrama de clases del diseño

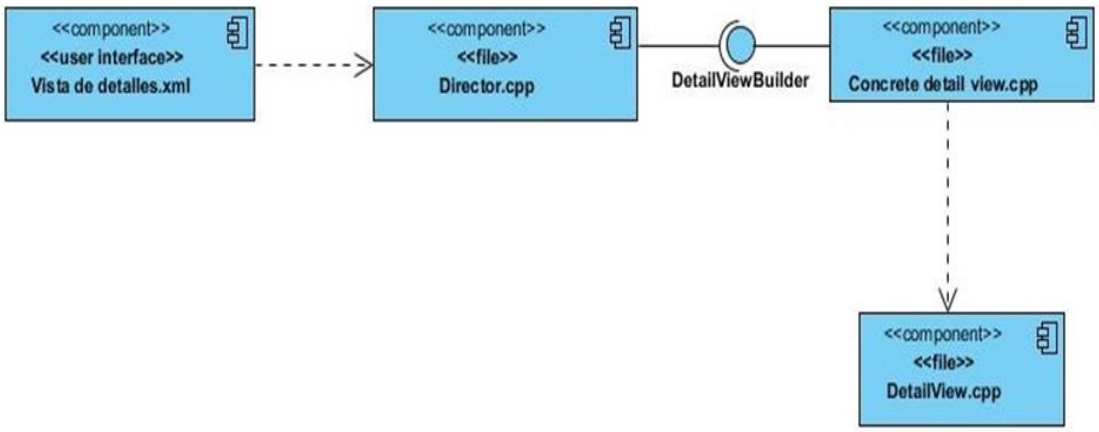


Figura 10 Diagrama de componentes

2.5.2 Prototipo de interfaz de usuario

Para la elaboración del prototipo se tuvieron en cuenta las características del módulo de planos y los requisitos funcionales planteados en la solución (Figura 11). Se empleó como herramienta el framework de desarrollo Qt 5.8.0 y como entorno de desarrollo se utilizó Qt Creator. El prototipo contiene las siguientes características visuales:

- Cualquier texto de los componentes de la aplicación se presenta capitalizado, es decir, con la primera letra en mayúscula.
- Los colores de fondo proporcionan el contraste adecuado para una cómoda visualización de los textos, por lo que se usan contrastes altos entre el fondo y la letra, el fondo de color claro y la letra de color oscuro.

- El nombre de la pantalla es el mismo que se muestra en las funcionalidades.
- En todo lugar que se presenta un texto, ya sea en etiquetas de navegación, botones, contenido, entre otros, no existen errores gramaticales ni ortográficos.

El prototipo debe permitir crear e insertar una vista de detalle de una parte específica de una vista base, proyectada, auxiliar o superpuesta. Además, permite configurar la escala, etiqueta identificativa, en donde se va a proyectar y forma de recortar.

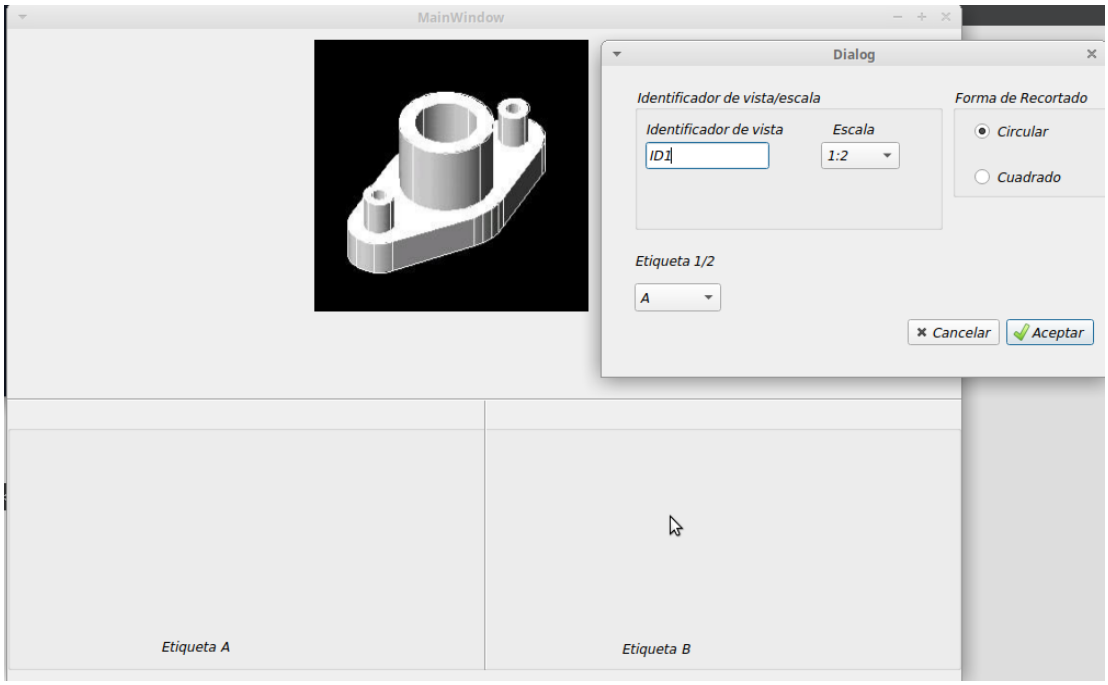


Figura 11 Prototipo de interfaz de usuario

Conclusiones Parciales

Después de realizado el análisis y el diseño de la funcionalidad de vistas de detalle se llegó a la siguiente conclusión: Con la obtención de los requisitos funcionales y no funcionales se satisface las necesidades planteadas en el objetivo de la investigación. Los artefactos ingenieriles generados de acuerdo a la metodología escogida, la propuesta arquitectónica y de diseño crearon las bases necesarias para la futura implementación de la propuesta de solución.

Capitulo III. Implementación y prueba

Introducción

En el presente capítulo se describen los principales detalles de la implementación del módulo. Se especifican las pruebas realizadas al módulo para validar su correcto funcionamiento con el fin de asegurar que el mismo cumpla con las funcionalidades y tenga la calidad requerida.

3.1 Implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios [20].

Para la implementación del módulo se tuvo en cuenta que, este dentro de su código debía tener las clases PlanoPlugin y PlanoState ya que estas van a permitir que el sistema AsiXmec reconozca el módulo y lo cargue, las cuales permitirán una correcta integración del módulo al sistema.

3.2 Estándar de codificación

Un estándar de codificación es un conjunto de reglas y acuerdos utilizados al escribir un código fuente en un lenguaje de programación particular. Este define varios aspectos de la creación y el mantenimiento de los códigos fuente de los programas. Por ejemplo, entre tales aspectos podemos nombrar las reglas de nomenclatura de variables, estilo de sangría, métodos de arreglo de corchetes, uso de espacios al formatear expresiones aritméticas, estilo de comentario, etc. Un estándar de codificación da una apariencia uniforme a los códigos escritos por diferentes ingenieros esto mejora la legibilidad y la mantenibilidad del código y también reduce la complejidad, ayuda en la reutilización del código y ayuda a detectar errores fácilmente, promueve buenas prácticas de programación y aumenta la eficiencia de los programadores [21].

QT y C++:

Todo el código fuente poseerá una codificación UTF-8

Cada subclase QObject (Figura 12) debe tener una macro Q_OBJECT

```
// in qclassname.h
#ifndef QCLASSNAME_H
#define QCLASSNAME_H

// public declarations

#endif // QCLASSNAME_H
```

```
// in qclassname_p.h
#ifndef QCLASSNAME_P_H
#define QCLASSNAME_P_H

// private declarations

#endif // QCLASSNAME_P_H
```

Figura 12 QObject Fuente:[21]

Normalizar los argumentos para señales + ranuras dentro de las declaraciones de conexión para obtener búsquedas más rápidas de señales/ranuras.

Siempre usar protectores de inclusión convencionales en encabezados de bibliotecas públicas y privadas:

```
#include <qstring.h>
// #include Qt stuff
// ...

#include <new>
// #include STL stuff
// ...

#include <limits.h>
// #include system stuff
// ...
```

Figura 13 Protectores Fuente:[21]

En los archivos de encabezado públicos, usar siempre este formulario para incluir encabezados Qt: #include <QtCore/qwhatever.h>.

En los archivos fuente, incluir primero encabezados especializados y luego encabezados genéricos.

Al usar el operador de signo de interrogación si los tipos devueltos no son idénticos, algunos compiladores generan código que falla en tiempo de ejecución (ni siquiera recibirá una advertencia del compilador):

3.3 Prueba

Las pruebas de software se realizan con el objetico de proporcional información sobre la calidad de un producto que se está desarrollando, estas son una actividad más en el proceso de control de calidad. Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software, dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo. Existen distintos modelos de desarrollo de software, así como modelos de pruebas. A cada uno corresponde un nivel distinto de involucramiento en las actividades de desarrollo. Luego de la implementación de la solución, se realizan un conjunto de actividades para verificar la calidad del producto y su cumplimiento con los requisitos definidos. El objetivo de esta fase es detectar y solucionar los errores que presenta el componente desarrollado, y perfeccionar la solución implementada [22].

Sommerville en su libro Ingeniería del software plantea que el proceso de prueba de software tiene dos objetivos distintos [23]**:**

• Demostrar al desarrollador y al cliente que el software satisface sus requisitos. Esto significa que debería haber al menos una prueba para cada requerimiento o característica que se incorporará a la entrega del producto.

• Descubrir defectos en el software en el que el comportamiento de este es incorrecto, no deseable o no cumple su especificación. La prueba de defectos está relacionada con la eliminación de todos los tipos de comportamientos del sistema no deseables, tales como caídas del sistema, interacciones no permitidas con otros sistemas, cálculos incorrectos y corrupción de datos.

La metodología AUP-UCI define tres etapas para la realización de las pruebas [14]**:**

• **Pruebas internas**: se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de pruebas ejecutables para automatizar las pruebas.

• **Pruebas de liberación:** Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

• **Pruebas de Aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

3.3.1 Métodos de prueba

Los métodos de pruebas definen estrategias para descubrir fallos en el sistema, como partes de estos Pressman en su 7ma edición [23], propone los siguientes:

Pruebas de caja blanca

Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que: garanticen que se ejercite por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; ejecuten todos los bucles en sus límites y con sus límites operacionales, y que se ejerciten las estructuras internas de datos para asegurar su validez. Estas pruebas se realizan al código fuente para asegurar que la operación interna se ajuste a las especificaciones [24].

Pruebas de caja negra

Las pruebas de caja negra también conocidas como pruebas funcionales o pruebas de entrada y salida, son las que se ejecutan sobre la interfaz del software. Mediante el uso de estas se examinan todas las funcionalidades. Estas tienen poca relación con el comportamiento interno del software. Estas pruebas permiten demostrar que las funciones del sistema sean operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta [25].

3.3.2 Estrategia de prueba

Una estrategia de prueba del software integra los métodos de diseño de casos de pruebas del software en una serie bien planeada de pasos que desembocará en la eficaz construcción del software. La estrategia proporciona un mapa que describe los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar estos pasos, y cuánto esfuerzo, tiempo y recursos se van a requerir. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el

diseño de casos de prueba, la ejecución de las pruebas y la agrupación y evaluación de los datos resultantes.

Pressman en la 6ta edición del libro “Ingeniería de software” Un enfoque práctico” propone cuatro niveles de prueba funcionales [26]**:**

Pruebas unitarias: son pruebas de caja blanca que se realizan con el objetivo de detectar errores de implementación en el componente desarrollado.

Pruebas de integración: verifican que cada componente desarrollado no presente errores cuando se integre con los demás.

Pruebas de validación: son pruebas de caja negra que se enfocan en la satisfacción de las necesidades del usuario.

Pruebas del sistema: son pruebas que confirman el correcto funcionamiento de las funciones desarrollas.

Niveles de pruebas no funcionales [26]:

Prueba de Interfaz de Usuario: en esta se evalúa la interfaz gráfica del usuario con respecto a este teniendo en cuenta su apariencia y si los datos corren correctamente.

Pruebas de Portabilidad: se realizan para verificar si un software / aplicación desarrollado se puede ejecutar en sistemas diferentes o en plataformas diferentes.

Pruebas de soportabilidad: está enfocada en asegurar que el sistema desarrollado funciona en diferentes configuraciones de hardware y software.

Para la validación de la propuesta de solución de la presente investigación, se definió una estrategia de prueba funcionales que tiene pruebas de unidad en correspondencia con uno de los 4 niveles antes mencionados y las pruebas no funcionales antes planteadas. Además, esta estrategia se diseñó teniendo en cuenta la metodología AUP-UCI, seleccionada para guiar el desarrollo de la aplicación.

3.3.3 Diseño de casos de prueba

Para validar los requisitos funcionales de la aplicación se diseñan casos de prueba a partir de las historias de usuario, con el objetivo fundamental de encontrar la mayor cantidad posible de deficiencias existentes en las funcionalidades implementadas.

Tabla 3.1 Diseño de casos de prueba # 1

Número del caso de prueba	1	
Nombre del caso de prueba	Crear Vista de detalle	
Descripción	Le permite al usurario crear una vista de detalle	
Precondiciones		
Fecha	20/11/2022	
Escenario	Descripción	Respuesta del Sistema
EC 1.1 Clik en la parte de pieza a mostrar	Seleccionar parte de la vista a mostrar	El sistema muestra una interfaz con las opciones de la viste a configurar.
EC 1.2 Clik en la opción “Aceptar”	Seleccionar opción “Aceptar”	El sistema muestra la parte de la vista seleccionada.
EC 1.3 Clik en la opción “Cancelar”	Seleccionar opción “Cancelar”	El sistema cierra la ventana de la vista.

Tabla 3.2 Diseño de casos de prueba #2

Número del caso de prueba	2	
Nombre del caso de prueba	Configurar escala	
Descripción	Le permite al usuario seleccionar la escala en que se mostrara la vista.	
Precondiciones		
Fecha	20/11/2022	
Escenario	Descripción	Respuesta del Sistema
EC 2.1 Opción	Seleccionar	Brinda la posibilidad de seleccionar el valor

seleccionar escala	“Escala”	de la escala.
EC 2.2 Modificar escala	Clik en la vista mostrada	Da la posibilidad de modificar la escala.

Tabla 3.3 Diseño de casos de prueba #3

Número del caso de prueba	3	
Nombre del caso de prueba	Crear etiqueta identificativa	
Descripción	Le permite al usurario crear una etiqueta a la vista	
Precondiciones		
Fecha	20/11/2022	
Escenario	Descripción	Respuesta del Sistema
EC 3.1 Opción “Aceptar”	Clik en la opción Aceptar.	Muestra una etiqueta por defecto(ID1).
EC 3.2 Opción “Identificador”	Escribir en el Line Edit.	Muestra la vista con el identificador escrito.
EC 3.3 Modificar Identificador	Clik en la vista a modificar.	Muestra una interfaz para modificar el identificador.

Tabla 3.4 Diseño de casos de prueba #4

Número del caso de prueba	4
Nombre del caso de prueba	Seleccionar espacio de proyección.
Descripción	Le permite al usurario seleccionar el espacio donde será mostrado la vista.
Precondiciones	

Fecha	20/11/2022	
Escenario	Descripción	Respuesta del Sistema
EC 4.1 Opción “Aceptar”	Clik en la opción “Aceptar”	Muestra la vista en el espacio(A).
EC 4.2 Opción “Etiqueta”	Seleccionar el “ComboBox” ”Etiqueta”	Muestra los espacios a seleccionar por el usuario para mostrar la vista.

Tabla 3.5 Diseño de casos de prueba #5

Número del caso de prueba	5	
Nombre del caso de prueba	Seleccionar la forma de recortar	
Descripción	Le permite al usurario seleccionar la forma en que se vera la vista .	
Precondiciones		
Fecha	20/11/2022	
Escenario	Descripción	Respuesta del Sistema
EC 5.1 Opción ” Aceptar”	Seleccionar Opción “Aceptar”	Muestra la vista en forma de circulo.
EC 5.2 Opción “Cuadrado”	Seleccionar Opción “Cuadrado”	Muestra la vista en forma de cuadrado.
EC 5.3 Modificar la forma de recortar	Clik en la vista a modificar	Le permite al usuario seleccionar la forma en que se vera la vista.

3.3.4 Pruebas unitarias

Las pruebas unitarias se aplican a un componente del software. Podemos considerar como componente (elemento indivisible) a una función, una clase, una librería. Estas pruebas las ejecuta el desarrollador, cada vez que va probando fragmentos de código o scripts para ver si todo funciona como se desea. Estas pruebas son muy técnicas. Por ejemplo, probar una consulta, probar que un fragmento de código envíe a imprimir un documento, probar que una función devuelva un flag [27].

Para la realización de las pruebas unitarias se utiliza el “Qt Test”, un framework para realizar pruebas unitarias a aplicaciones y bibliotecas basadas en Qt. El “Qt Test” proporciona todas las funcionalidades comúnmente encontradas en los framework de pruebas, así como extensiones para probar interfaces gráficas de usuario [28]. Las pruebas fueron ejecutadas sistemáticamente, cada vez que se terminaba de implementar cada una de las iteraciones.

```
#include <QtTest>
#include <QTestKeyClicksEvent>

// add necessary includes here

class Qtest : public QObject
{
    Q_OBJECT     incomplete result type 'QString' in function definition

public:
    Qtest();
    ~Qtest();

private slots:
    void test_case1();
    void ClickableLabel::ClickableLabel(QWidget *parent) : QLabel(parent);
    void ClickableLabel::~ClickableLabel();
};


void Qtest::test_case1()
{
  ClickableLabel::mouseReleaseEvent(QMouseEvent *ev)     use of undeclared

        const QPoint pos = ev->pos();
        emit clicked(pos);
}

QTEST_APPLESS_MAIN(Qtest)                                  use of undeclare

#include "tst_qtest.moc"
```

Figura 14 Ejemplo de código para pruebas unitarias usando QtTest

3.3.5 Pruebas Internas

Las pruebas internas son aquellas donde se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Este tipo de pruebas se basa en las funcionalidades de un sistema que se describen en la especificación de

requisitos, es decir, lo que hace el sistema [29].

Para la realización de estas pruebas se utilizaron los diseños de casos descritos en el epígrafe 3.3.3.

Al concluir cada una de las iteraciones planificadas para el desarrollo de la propuesta de solución, en la Figura 15 se muestra el resultado obtenido:

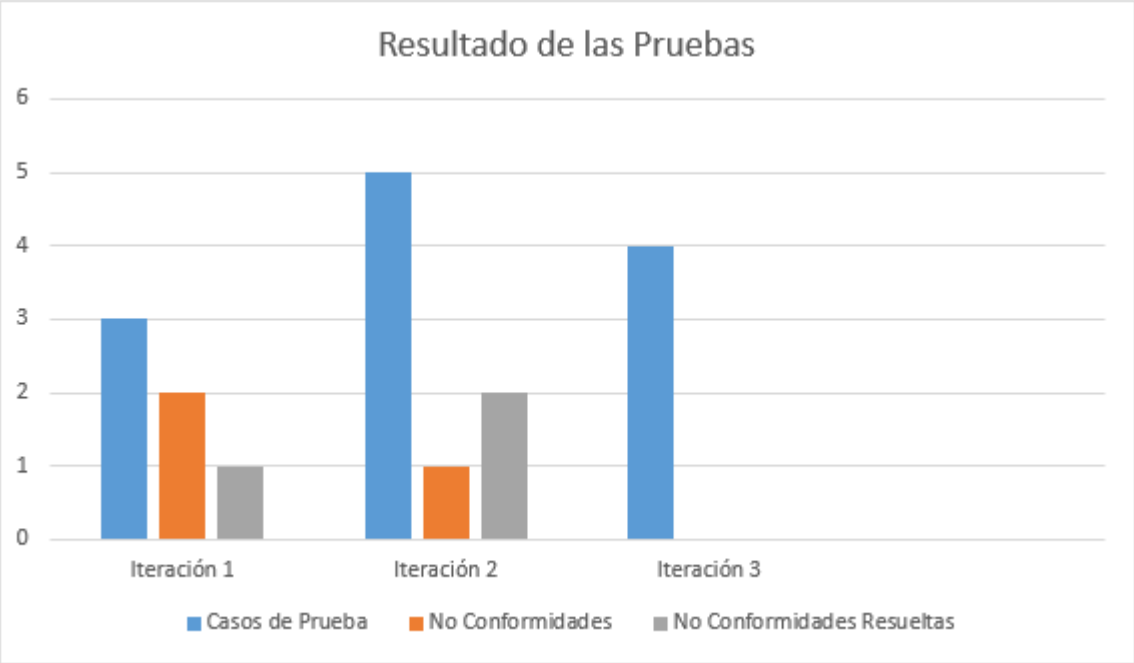


Figura 15 Resultado de las pruebas realizadas durante las iteraciones de prueba

En la primera iteración se realizó tres casos de pruebas en la cual se detectaron dos no conformidades y se resolvió una; en la segunda iteración se realizó cinco casos de pruebas en las cuales se detectaron una no conformidad y se resolvió 2; en la tercera iteración se realizó cuatro casos de pruebas en la cual no se detectaron no conformidades en el sistema.

Conclusiones Parciales

La definición de los estándares de codificación permitió mejorar el entendimiento del código perteneciente módulo. Al trazar la estrategia de prueba se definió una serie de pasos que nos dan como resultado una correcta construcción del sistema, entre los que se incluyen los tipos de prueba que se le realizaron al software. La utilización de la técnica de caja negra (pruebas de aceptación) y pruebas de caja blanca (pruebas unitarias), se permitió identificar un total de 3 no conformidades en el módulo, las cuales fueron corregidas garantizando una mejor calidad de la solución.

Conclusiones Generales

Como resultado de la investigación y los resultados obtenidos, se concluye:

- A partir del análisis de los sistemas CAD similares se decidió que la propuesta de solución debía tener elementos como crear una etiqueta identificativa, seleccionar la escala de la vista, modificar la escala de la vista, entre otros.
- El empleo de la metodología AUP-UCI en su escenario 4 y las herramientas y tecnologías Qt Creator y Qt Framework posibilitó obtener la documentación necesaria para llevar a cabo las actividades de implementación.
- El desarrollo del módulo permite crear una vista de detalle a partir de una vista ya existente, cumpliendo con los requisitos funcionales planteados y el objetivo de la investigación.
- Este módulo posibilitó tener una funcionalidad de vista de detalle para el sistema AsiXmec.
- Para garantizar la calidad del módulo desarrollado se realizaron las validaciones necesarias obteniendo resultados satisfactorios en cada una, ya que todas las no conformidades encontradas fueron corregidas.

Recomendaciones

A partir del trabajo realizado y luego de haber analizado los resultados obtenidos se sugiere el siguiente elemento:

- Incluir el módulo de vista de detalle al sistema AsiXmec.

Referencias Bibliográficas

[1] «Qué es CAD, para qué sirve y qué ventajas tiene», *Integral Innovation Experts Blog*, 20 de agosto de 2019. https://integralplm.com/blog/2019/08/20/que-es-cad/ (accedido 26 de noviembre de 2022).

[2] «Modulo De Dibujo Tecnico», *calameo.com*. https://www.calameo.com/read/0059563993c82e0a785e8 (accedido 26 de noviembre de 2022).

[3] «Acerca de las vistas de detalle | AutoCAD 2019 | Autodesk Knowledge Network». https://knowledge.autodesk.com/es/support/autocad/learn-explore/caas/CloudHelp/cloudhelp/2019/ESP/AutoCAD-Core/files/GUID-11B81B84-3ACC-46F5-9E6D-0B75C78B3A9F-htm.html (accedido 29 de noviembre de 2022).

[4] «Creación de vistas de recortes y de detalles con la función Sin contorno - 2017 - Ayuda de SOLIDWORKS». https://help.solidworks.com/2017/spanish/solidworks/sldworks/t_creating_crop_view_no_outline.htm (accedido 29 de noviembre de 2022).

[5] «Dialnet-ModuloParaAutomatizarElDisenoDeEngranajesDeTornill-8589945-1.pdf».

[6] «AutoCAD», *Wikipedia, la enciclopedia libre*. 13 de octubre de 2022. Accedido: 26 de noviembre de 2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=AutoCAD&oldid=146596682

[7] «Autodesk Inc., ¿qué es Autodesk? | Espacio BIM». https://www.espaciobim.com/autodesk (accedido 26 de noviembre de 2022).

[8] «Authentic Design Experience | SOLIDWORKS – Dassault Systèmes». https://www.3ds.com/es/productos-y-servicios/solidworks/ (accedido 26 de noviembre de 2022).

[9] «C++», *Wikipedia, la enciclopedia libre*. 8 de noviembre de 2022. Accedido: 26 de noviembre de 2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=C%2B%2B&oldid=147177698

[10] «Open Cascade Technology», *Wikipedia*. 15 de noviembre de 2022. Accedido: 26 de noviembre de 2022. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Open_Cascade_Technology&oldid=1122056302

[11] «Qt (software)», *Wikipedia*. 21 de noviembre de 2022. Accedido: 26 de noviembre de 2022. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Qt_(software)&oldid=1123068352

[12] «Qt Creator», *Wikipedia*. 8 de noviembre de 2022. Accedido: 26 de noviembre de 2022. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Qt_Creator&oldid=1120741027

[13] «Doxygen», *Wikipedia, la enciclopedia libre*. 24 de enero de 2022. Accedido: 26 de noviembre de 2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Doxygen&oldid=141190373

[14] «Variación de AUP para la UCI (AUP-UCI) - Metodologías tradicionales». https://1library.co/article/variaci%C3%B3n-aup-uci-aup-uci-

metodolog%C3%ADas-tradicionales.zkwgn934 (accedido 28 de noviembre de 2022).
[15] Adrian Peña Peñate, «AsiXMec: Aplicación cubana para el Diseño e Ingeniería Asistida por Computadora», 2021. [En línea]. Disponible en: : https://www.3dcadportal. com / asixmec - aplicacion - cubana - para - el - diseno - e - ingenieria - asistida - por - computadora.html
[16] «Educción de requisitos», *Wikipedia, la enciclopedia libre*. 22 de octubre de 2019. Accedido: 26 de noviembre de 2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Educci%C3%B3n_de_requisitos&oldid=120660113
[17] «Arquitectura de Microkernel». https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/microkernel (accedido 29 de noviembre de 2022).
[18] «Builder». https://reactiveprogramming.io/blog/es/patrones-de-diseno/builder (accedido 28 de noviembre de 2022).
[19] «Builder Pattern: soluciones de software más rápidas con el patrón Builder». https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/patron-de-diseno-builder/ (accedido 29 de noviembre de 2022).
[20] «Implementación», *Wikipedia, la enciclopedia libre*. 15 de octubre de 2022. Accedido: 28 de noviembre de 2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Implementaci%C3%B3n&oldid=146632507
[21] «Qt Coding Style - Qt Wiki». https://wiki.qt.io/Qt_Coding_Style (accedido 26 de noviembre de 2022).
[22] «Pruebas de software - Wikipedia, la enciclopedia libre». https://es.wikipedia.org/wiki/Pruebas_de_software (accedido 26 de noviembre de 2022).
[23] «Ingenieria del Software 7ma. Ed. - Ian Sommerville.pdf».
[24] «Pruebas de caja blanca», *Wikipedia, la enciclopedia libre*. 20 de julio de 2022. Accedido: 26 de noviembre de 2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Pruebas_de_caja_blanca&oldid=144870377
[25] «Caja negra (sistemas)», *Wikipedia, la enciclopedia libre*. 5 de octubre de 2022. Accedido: 26 de noviembre de 2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Caja_negra_(sistemas)&oldid=146399841
[26] «toaz.info-ingenieria-del-software-roger-s-pressman-6ta-edicion2pdf-pr_c28d1f91ab0590ebbcba9b97735c314b.pdf».
[27] «Prueba unitaria», *Wikipedia, la enciclopedia libre*. 27 de junio de 2022. Accedido: 26 de noviembre de 2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Prueba_unitaria&oldid=144441113
[28] «Qt Test Overview | Qt Test 6.4.1». https://doc.qt.io/qt-6/qtest-overview.html (accedido 28 de noviembre de 2022).
[29] «JOSE_MANUEL_SANCHEZ_PENO_3.pdf».