Facultad 5

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

Componente para la interpretación y visualización de cartas dinagráficas.

Autor(es): Yurelys Naranjo Palomino,

Hansel Albellar Matos.

Tutor: Ing**.** José Ernesto Carreño Bueno.

Co-tutor: Ing**.** Frank Rufino Nápoles.

“La Habana, junio, 2015”

“Año 57 de la Revolución”

Declaración de Autoría

DECLARACIÓN DE AUTORÍA:

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

Para que así conste firmamos la presente a los __ días del mes _____ del año 2015.

Autor

_____

Yurelys Naranjo Palomino

Autor

_____

Hansel Albellar Matos

Tutor

_____

José Ernesto Carreño Bueno

Co-tutor

_____

Frank Rufino Nápoles

Datos de Contacto

DATOS DEL CONTACTO

Tutor: José Ernesto Carreño Bueno

Categoría Científica: Ingeniero

Especialidad: Ingeniero en Ciencias Informáticas

Correo Electrónico: jecarreno@uci.cu

Categoría docente:-

Años de experiencia: 2

Años de graduado: 2

Co-tutor: Frank Rufino Nápoles

Categoría Científica: Ingeniero

Especialidad: Ingeniero en Ciencias Informáticas

Correo Electrónico: frufino@uci.cu

Categoría docente:

Años de experiencia: 4

Años de graduado: 4

Agradecimientos

Agradecimientos

A mi madre Nora, por dedicar su vida a sus hijas y ser un ejemplo de mujer intelectual ante la familia y la sociedad.

A mi padre Camilo, por servir de ejemplo de perseverancia y liderazgo para mí.

A mi hermana Yudelsy, por sentirla siempre como mi segunda madre y haberme guiado hacia la superación.

A mis abuelas y abuelos que siempre los he sentido conmigo, te extraño Ángela, Marina, Elvira, Mario, Francisco, y Hermes y mi amada Nelly que me sostiene y me resguarda.

A mi bella sobrina que fue el rayo de luz en nuestra familia,cuando todo se volvía oscuro.

A mis tías y tíos que me respetan y apoyan en cada decisión, en especial a mi querida Nelvy.

A mis primas y primos en especial a mis incondicionales Daniuska y Yurania que siempre han sido hermanas para mí.

A mi segundo padre Fily por su amor y dedicación.

A mi novio y su familia por ser mi mano derecha, por haberme enseñado la vida desde un punto diferente.

A mis amigos Eliizabet, Manuel, Leiser, Orlando, Hubert y Leyanet por haber estado siempre ahí con la solución a mis locuras.

A mis amigos Lesyania, Armandito, Isis y Leyanet, por compartir momentos increíbles a su lado.

A mi compañero de tesis y amigo por siempre tener esa actitud positiva al compartir este sueño junto a mí.

A mis compañeros de aula, a Teijón, Livan, Julio Cesar, Alvaro y Yosvany.
A mis tutores y profesores del proyecto SCADA por haberme guiado durante la investigación y el desarrollo de la tesis.

De Yurelys

A mis padres por ayudarme a ser el hombre que soy, y ser ejemplo de amor para mí.
A mi hermano por estar siempre a mi lado en las buenas y malas.
A mis abuelos por haberme enseñado que la vida es para vivirla minuto a minuto con intensidad y dando lo mejor de mí.
A mis tíos y tías por siempre estar pendientes de mí.
A mi compañera de tesis y amiga por ser mi mano derecha, siempre con esa sonrisa en su rostro.
A mis amigos de la UCI, Leonar, Reinier, Peru, Nolberto, Eulogio, por compartir tantos momentos buenos conmigo.
A toda mi familia en general , mis amigos y mis compañeros de la vida universitaria.
A mis tutores por siempre apoyarnos en la realización de nuestro trabajo de diploma.

De Hansel

Dedicatoria

Dedicatoria

A mi madre que es el sostén de mi vida.

A mi sobrina para que siga la tradición familiar de superación.

A mi tío para su pronta recuperación.

A mi familia por estar siempre a mi lado y ayudarme a realizar mi sueño de estudiar.

Yurelys

A mis padres por estar siempre conmigo brindándome amor.

A mi abuelo que siempre está en mi corazón.

A mi familia por apoyarme incondicionalmente.

Hansel

Resumen

Las cartas dinagráficas mediante el uso de programas computacionales, permiten detectar problemas con la bomba de petróleo, muestran el comportamiento de la misma y permiten mejorar la eficiencia de cualquier sistema petrolero previendo accidentes estructurales en el pozo. Se puede encontrar cartas de superficie o de fondo, su diferencia consiste en el lugar donde se adquieren los datos a procesar. En el presente trabajo se desarrolla y se incluye un componente de interpretación y visualización de cartas dinagráficas en la Interfaz Hombre-Máquina (HMI) del sistema de Adquisición, Supervisión y Control de Datos (SCADA) del Centro de Informática Industrial (CEDIN) de la facultad 5 de la Universidad de las Ciencias Informáticas (UCI).

Palabras clave: cartas dinagráficas; componente; sistema petrolero; supervisión; pozo

Abstract

The dinagraphics cards allow using computer programs to detect problems with the oil pump, shows the behavior of the same and improves the efficiency of any oil system structural preventing accidents in the well. Can be founded two kinds of cards, surface cards and background cards, the difference between this two kinds of card are where the data to be processed are acquired. In this work develops the inclusion of a component of interpretation and display of dinagraphics cards in the Human-Machine-Interface (HMI) system of Supervisory Control and Data Acquisition (SCADA) of Industrial Computing Center (CEDIN) of the Faculty 5 of the UCI.

Keywords: *component; dinagraphics cards; oil system; oil system; supervision; well*

Contenido

Indice de Contenidos

Índice de ilustraciones

Indice de Tablas

Indice de Tablas

INTRODUCCIÓN

Durante la segunda mitad del siglo veinte y lo que va del presente siglo la humanidad ha sido testigo de un desarrollo industrial sin precedentes en su historia. Debido a la complejidad de los procesos que se llevan a cabo en las industrias modernas y para tener un control de lo que ocurre dentro de la misma se han desarrollado los sistemas SCADA, en inglés *Supervisory Control and Data Adquisition*, en español, Control, Supervisión y Adquisición de Datos.

El papel fundamental de los sistemas SCADA es la obtención de información proveniente de dispositivos de campo dentro de la industria, esta información es transmitida a una estación que supervisa los procesos y administra dicha información. Además ellos están en condiciones de prevenir y alertar situaciones inesperadas producidas por esos dispositivos dentro de la industria.

La Universidad de las Ciencias Informáticas (UCI) específicamente el Centro de Informática Industrial (CEDIN) de la facultad 5 desarrolla sistemas de este tipo, ejemplo de esto es el SCADA GALBA (Guardián del Alba) desplegado con éxito en varias industrias Venezolanas en el sector petrolero.

La producción petrolera requiere cada día instrumentos de análisis más sofisticados que le permitan mantener su ritmo y sus costos. Una de las herramientas más usadas en el diagnóstico operacional de pozos es la carta dinagráfica de fondo y superficie. Estas cartas son curvas de esfuerzo por desplazamiento de la cabilla de la bomba, pero calculadas en el fondo del pozo a partir de datos tomados en la superficie. Usadas para diagnosticar el estado estructural del pozo y alargar la vida de la bomba.

El sistema SCADA GALBA mediante el driver Modbus ELAM extrae las cartas, pero no posee el sistema que interprete y visualice las mismas, actualmente para analizar este tipo de carta existen ya programas realizados pero ninguno específicamente para SCADA GALBA, podemos citar el XDIAG para Windows, que depende de plataformas privadas en las cuales hay que pagar licencia por su adquisición.

Por tanto, la **situación problemática** es la siguiente:

- Hay poco conocimiento del estado estructural de los pozos.
- Existen dificultades para prevenir posibles incidentes de derrumbe o rompimiento de la cabilla.
- Existen limitaciones para contribuir a garantizar la calidad del producto extraído en cuanto a parámetros de contaminación.

- Pueden ocasionarse accidentes debidos al rompimiento de la cabilla lo que implica inversión monetaria adicional y que no se maximice la vida útil de la bomba ya que atentan contra el correcto funcionamiento y la eficiencia del sistema.

Luego, a partir de la situación problemática planteada se permite formular el siguiente **problema científico**: ¿Cómo contribuir a la interpretación y visualización de las cartas dinagráficas en el SCADA GALBA?

Para darle solución al problema se propone como **objetivo general:** Desarrollar un componente que interprete y visualice las cartas dinagráficas en el SCADA GALBA..

El problema planteado define el siguiente **objeto de estudio**: Proceso de interpretación y visualización de cartas dinagráficas.

Todo lo cual precisa como **campo acción**: Proceso de interpretación y visualización de cartas dinagráficas brindadas por el driver Modbus ELAM del SCADA GALBA.

Con vistas a dar cumplimiento al objetivo general se tienen las siguientes **tareas de investigación:**

- Elaboración del marco teórico de la investigación a partir del estado del arte existente actualmente sobre el tema.
- Realización de la investigación de las características propias de las cartas dinagráficas para definir los tipos de cartas existentes.
- Selección de la metodología y tecnología de desarrollo acorde con las políticas de la línea de HMI para los componentes gráficos.
- Diseño de la aplicación utilizando las herramientas de desarrollo de software.
- Levantamiento de los requisitos no funcionales del sistema.
- Integración en el SCADA del componente gráfico.
- Implementación de componentes de interpretación y visualización de cartas dinagráficas.
- Pruebas al componente.

Para darle cumplimiento a las tareas de investigación, fueron utilizados varios **métodos científicos de investigación**, entre los que se destacan como **métodos teóricos**:

- **Analítico-Sintético**: Utilizado para identificar los principales fundamentos teóricos, así como las características fundamentales de las cartas dinagráficas y los elementos relacionados, a través del estudio de documentación recopilada.

- **Modelación**: Utilizado para la elaboración de diagramas que representan la estructura, relaciones internas y características del objeto de investigación. Creando abstracciones que permiten explicar la realidad.

Como **método empírico** se utilizó:

- **Consulta de información en todo tipo de fuente:** Lo cual permite la elaboración del marco teórico de la investigación.
- **Experimental**: Empleado en la realización de pruebas, donde se crean las condiciones propicias para verificar el correcto funcionamiento del componente desarrollado en la presente investigación.
- **Entrevista**: Para conocer lo relacionado con cartas dinagráficas con los principales expertos en el tema.

El presente documento está estructurado de la siguiente manera:

Capítulo 1: Fundamentación teórica. En este capítulo se elabora el marco teórico de la investigación mediante la definición de conceptos y características fundamentales de las cartas dinagráficas. Además se definen las tecnologías y herramientas que se utilizarán para realizar el diseño, y posterior implementación de la solución propuesta.

Capítulo 2: Análisis y diseño. Se lleva a cabo el modelado de las funcionalidades y el diseño de las clases que se utilizarán en la implementación.

Capítulo 3: Implementación y prueba. Se realiza la implementación del componente de acuerdo a las funciones y clases diseñadas previamente. Además se muestran las pruebas realizadas al mismo y los resultados obtenidos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En la actualidad el uso de las cartas dinagráficas para pozos es común en la industria petrolera debido a las ventajas que esta proporciona en la predicción de posibles problemas relacionados el funcionamiento del bombeo, mediante una correcta visualización e interpretación se logra diagnosticar rápidamente para ofrecer soluciones. En el siguiente capítulo se muestran características, conceptos y definiciones de los elementos relacionados con las cartas dinagráficas, su visualización e interpretación, se describen las principales funciones del SCADA, y los módulos que lo estructuran, haciendo énfasis en la visualización de los componentes. Además se mencionan las herramientas y las tecnologías utilizadas para el desarrollo de la solución propuesta.

1.1 Conceptos asociados al dominio del problema

Para la comprensión del problema se deben dominar conceptos como sistemas SCADA, HMI, cartas dinagráficas, Modbus ELAM, entre otros que se presentan a continuación:

1.1.1 Sistemas SCADA

El sistema SCADA (Supervisión, Control y Adquisición de Datos) del inglés *Supervisory Control and Data Acquisition* son aplicaciones de software creadas para controlar y supervisar procesos a distancia, basados en la adquisición de datos de procesos remotos. Las funciones principales son:

- Supervisión: El operador podrá observar desde el monitor la evolución de las variables de control, como cambios que se produzcan en la operación diaria de la planta, lo que permite dirigir las tareas de mantenimiento y estadística de fallas.

- Control: Mediante el sistema se puede activar o desactivar los equipos remotamente (por ejemplo abrir válvulas, activar interruptores, prender motores, etc.), de manera automática y también manual.

El operador puede ejecutar acciones de control y podrá modificar la evolución del proceso en situaciones irregulares que se generen.

- Adquisición de datos: Recolectar, procesar, almacenar y mostrar la información recibida en forma continua desde los equipos de campo. (Ledesma Marcalla, y otros, 2010)

Un SCADA se compone de bloques o módulos independientes que intervienen en su funcionamiento y permiten las actividades de supervisión, control y adquisición.

Entre los módulos que componen un SCADA están:

- Bases de Datos Históricos: Es el encargado de almacenar la información recibida desde el campo, así como la sucesión de alarmas y eventos. Esta información es de vital importancia para realizar cualquier tipo de análisis posteriores como diagnósticos o reportes. Debe permitir a los usuarios y ejecutivos el análisis rápido de información histórica, la que debe estar organizada según patrones de comportamiento de las plantas y dar datos relevantes de todo el proceso industrial.
- Comunicaciones o Middleware: Este módulo representa la capa de software encargada de la comunicación entre los diferentes procesos distribuidos de mediano y alto nivel. Enlaza todos los componentes del sistema y tiene relación directa con el canal de comunicación. Su arquitectura es compleja en la mayoría de los casos, pero presenta pequeñas interfaces que garantizan rapidez y robustez en la transmisión de mensajes. Es uno de los módulos horizontales al sistema y sus cambios suelen afectar el funcionamiento general del mismo.
- Seguridad: Permite a los usuarios autenticarse en el sistema, y de esta forma acceder solo a los recursos que tiene asignado su rol. Posee herramientas para la protección ante ataques piratas, fallos eléctricos, problemas de red, entre otros. Su objetivo es garantiza la accesibilidad, integridad y confidencialidad de la información manejada por el sistema, lo cruza horizontalmente, y se asocia de forma discreta, a todos los módulos del sistema.
- Procesamiento de Datos: Este módulo representa el "centro neurológico" del sistema, el cual supervisa y recoge la información del resto de las subestaciones, bien sean otros ordenadores conectados (en sistemas complejos) a los instrumentos de campo o directamente sobre dichos instrumentos.

- Ambiente de configuración: Permite configurar varios procesos o parte de ellos. En este módulo se gestionan las variables, configuración de manejadores y comandos internos del sistema. Sirve como capa abstracta que garantiza la optimización y coherencia entre los componentes, permite la direccionalidad mutua entre elementos internos y les ofrece una interfaz sencilla para su comunicación.
- Interfaz Hombre-Máquina o HMI: Interfaz que comunica al sistema con sus operadores.

1.1.2 Módulo Interfaz Hombre-Máquina

Una interfaz HMI en inglés *Human Machine Interface*, en español Interfaz Hombre Máquina, es un sistema que presenta datos a un operador y a través del cual éste controla un determinado proceso que ocurre en el campo. Las señales de los procesos son trasladadas al HMI por medio de dispositivos como tarjetas de entrada/salida en la computadora, controladores lógicos programables (PLC, por sus siglas en inglés), unidades terminales remotas de entrada y salida o manejadores. Todos estos dispositivos deben tener una comunicación que entienda el HMI. (HMI,2007)

1.1.3 Cartas Dinagráficas

Una carta dinagráfica es un registro de cargas versus posición a lo largo de la barra pulida en un instante durante un ciclo de bombeo. Estas permiten mediante el uso de programas computacionales detectar problemas con la bomba de subsuelo, el comportamiento de los equipos de superficie y al mismo tiempo permite mejorar la eficiencia del sistema.

Estas cartas son curvas de esfuerzo versus desplazamiento de la cabilla de la bomba pero calculadas en el fondo del pozo a partir de datos tomados en la superficie. Para este cálculo uno de los métodos más usados es el de la solución de la ecuación de onda de la cabilla, método desarrollado por Gibbs en los años sesenta. (Meza R, y otros, 2011)

1.1.4 Fase de Interpretación de la Cartas Dinagráficas

La interpretación puede ser cualitativa y cuantitativa aportando cada una de ellas información importante en el análisis de cada carta dinagráfica. El componente propuesto nos mostrará la fase de la interpretación cualitativa.

- Interpretación cualitativa

Con la interpretación cualitativa se aporta información de las condiciones o estado de los equipos de subsuelo, entrada de fluido a la bomba, esfuerzo sobre las cabillas y condiciones de producción del pozo. Este tipo de interpretación se basa en el análisis visual de la representación de una carta dinagráfica. (Meza R, y otros, 2011)

- Patrones teóricos para el análisis cualitativo a las cartas dinagráficas.

Representan patrones teóricos, los cuales sirven para el operador guiarse al momento de realizar un análisis cualitativo a las cartas dinagráficas y ambas especifican formas de cartas dinagráficas de superficie con el respectivo problema que se presente de acuerdo a cada una de las formas. (Meza R, y otros, 2011).

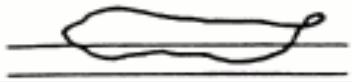
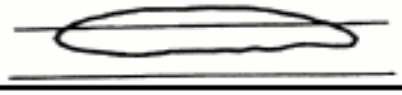
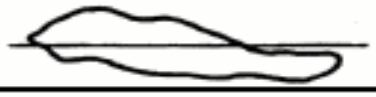
CARTA	EXPLICACIÓN
	BOMBA PEGANDO EN LA CARRERA ASCENDENTE SIN SALIR DE LA ZAPATA
	LA ARENA ATASCA O CASI ATASCA EL PISTÓN EN LA BOMBA
	PISTÓN DE LA BOMBA GOLPEANDO EN ARENA
	VÁLVULA FIJA PESCADA EN BOMBA DE TUBERÍA
	EL NIVEL DE FLUIDO COINCIDE CON LA VÁLVULA FIJA
	POZO BOMBEANDO COMPLETO (PUMPING - OFF)
	LÍNEA DE CARGA MOSTRANDO LA VÁLVULA VIAJERA MALA
	LÍNEA DE CARGA MOSTRANDO LA VÁLVULA FIJA MALA
	TUBO DE SUCCIÓN TAPADO POR ARENA
	PISTÓN EROSIONADO O GASTADO. EL FLUIDO ESTA PASANDO ENTRE EL PISTÓN Y LA CAMISA
	ALARGAMIENTO DE LAS CABILLAS DEBIDO A OBSTRUCCIONES EN LA LÍNEA DE FLUJO
	ALARGAMIENTO DE LAS CABILLAS DEBIDO A OBSTRUCCIONES EN LA LÍNEA DE FLUJO

Ilustración 1 Patrones teóricos para el análisis cualitativo

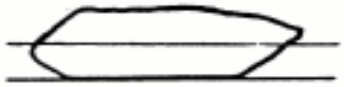
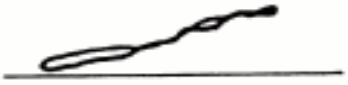
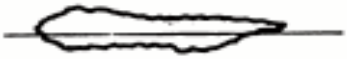
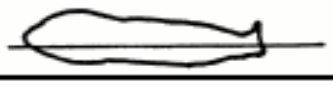
CARTA	EXPLICACIÓN
	BOMBEANDO NORMALMENTE
	BOMBA MANEJANDO MUCHO GAS
	POZO AGITADO. FLUIDO PRODUCIDO CON MUY ALTO RGL. LAS VÁLVULAS VIAJERA Y FIJA PERMANECEN ABIERTAS
	BOMBEO DE CRUDO MUY VISCOSO
	CABILLAS PARTIDAS O DESCONECTADAS
	VÁLVULA VIAJERA DAÑADA
	VÁLVULA FIJA CON FUGAS
	VÁLVULA FIJA CORTADA
	PISTÓN PEGADO
	CAJA DE ENGRANAJE DEL BALANCÍN DAÑADA
	PISTÓN GOLPEADO ABAJO
	PISTÓN GOLPEADO ARRIBA
	BOMBA DE CABILLAS SALIENDOSE DE LA ZAPATA AL FINAL DE CARRERA ASCENDENTE Y REENTRANDO EN LA DESCENDENTE

Ilustración 2.Patrones teóricos para el análisis cualitativo

- Interpretación cuantitativa

Para la interpretación cuantitativa se usan cálculos matemáticos que proporciona información de las cargas de la barra pulida; por lo que pueden obtenerse valores como la carga máxima sobre la barra pulida, carga mínima sobre la barra pulida, elongación de las cabillas, cargas de las cabillas en el fluido, cargas de la cabilla en el aire, flotabilidad de las cabillas en el fluido, carga en la válvula viajera, carga en la válvula fija, carga de contra balance, torque máximo de la unidad de bombeo, variación de torque con el recorrido de la barra pulida y la potencia del motor. (Meza R, y otros, 2011)

1.1.5 Bombeo mecánico

El bombeo mecánico es el método de producción primaria mediante elevación artificial del fluido que se encuentra en el pozo y que por falta de energía no puede surgir a superficie. Consiste en una bomba de subsuelo de acción reciprocante que es abastecida con energía transmitida a través de una sarta de varillas (cabillas). La energía proviene de un motor eléctrico o de combustión interna, la cual moviliza a una unidad de superficie mediante un sistema de engranaje y correas.

El bombeo mecánico es un procedimiento de succión y transferencia casi continua del petróleo hasta la superficie. La unidad de superficie imparte el movimiento de sube y baja a la sarta de varillas de succión que mueve el pistón de la bomba, colocada en la sarta de producción, a cierta profundidad del fondo del pozo. (2012)

1.1.6 Cabillas

Se pueden encontrar diferentes tipos de cabillas; convencionales o continuas, las primeras, sus longitudes pueden ser de 25 o 30 pies, utilizando niples de cabillas (tramos de cabillas de menor longitud), en los casos que ameriten para obtener la profundidad de asentamiento de la bomba, otros elementos adicionales de la sarta de cabillas podrían ser una barra (Sinker Bar), diseñado para adicionar peso al colocar en la parte inferior de la barras de peso es de 1 ½ a 2 pulgadas. En pozos productores de crudo pesado; donde se crea una especie de colchón que aumenta el efecto de flotación de las cabillas durante su carrera descendiente, dificultando el desplazamiento del pistón dentro del barril de la bomba, con una consecuente

disminución de la eficiencia volumétrica de la bomba, es ventajoso utilizar barra de peso en la sarta de cabillas, ya que facilita el desplazamiento de crudo viscoso, al mantener tensión en la sarta de cabillas. (Meza R, y otros, 2011)

Las segundas cabillas son; Electra, continuas, fibra de vidrio dentro de las cuales las más usadas son las cabillas continuas, su elongación es 3.8 veces mayor que las cabillas de acero para la igual carga y diámetro.

1.1.7 Modbus ELAM

El Protocolo Modbus extendido también conocida como ELAM del inglés Extended Lufkin Automation Modbus) es un protocolo propietario de Lufkin Automation que añade varias características para mejorar aún más el protocolo Modbus estándar. Este protocolo se utiliza para el propósito de comunicarse con controladores de Lufkin Automation y otros dispositivos compatibles. En nuestro centro se utiliza una versión creada por él mismo, del protocolo, que es configurado, para pedir que extraiga los datos en un formato específico, diseñado previamente, que será el mismo que se usará para configurar la carta en el componente, de esta manera se garantiza su concordancia. A continuación se muestra un ejemplo del formato, en que se extraen las cartas que fue utilizado para interpretar.

Register	# Regs	F	L	A	NAME	DESCRIPTION
32669	2035			X	5 Surface Card Buffer	5 Card Buffer Stored Cards - Top Card is Most Recent Shutdown Cards - Top Card is Least Recent

Each Card is Organized As:

	#	F	L	A	Name	Description
Header	2		X		TimeStamp	Seconds Since Jan 1, 1970
	1				Num Points/Shutdown Cause	Low Byte - Number of Points *High Byte - Shutdown Cause
	1				Scaled Max Load	Scaled Max Load Of Card (lbs.)
	1				Scaled Min Load	Scaled Min Load Of Card (lbs.)
	1				Stroke Length	Stroke Length x 100 (Inches)
	1				Stroke Period	Stroke Period x 100 (seconds)
Card				X	200 Point Pairs**	
	1				Position x 100	Scaled Position x 100 (in.)
	1				Scaled Load	Scaled Load (lbs.)

Ilustración 3 Carta dinagráfica de superficie.

34704	1045			X	5 Pump Card Buffer	5 Card Buffer Stored Cards - Top Card is Most Recent Shutdown Cards - Top Card is Least Recent

Each Card is Organized As:

	#	F	L	A	Name	Description
Header	2		X		TimeStamp	Seconds Since Jan 1, 1970
	1				Scaled Max Load	Scaled Max Load Of Card (lbs.)
	1				Scaled Min Load	Scaled Min Load Of Card (lbs.)
	1				Num Points/Shutdown Cause	Low Byte - Number of Points *High Byte - Shutdown Cause
	1				Gross Stroke	x 100 (in.)
	1				Net Stroke	Net Stroke Length x100 (in.)
	1				Pump Fillage	x 100 (%)
	1				Fluid Load	Fluid Load (lbs.)
Card				X	100 Point Pairs*	
	1				Position x 100	Scaled Position x 100 (in.)
	1				Scaled Load	Scaled Load (lbs.)

Ilustración 4 Carta dinagráfica de fondo

1.2 Tecnologías y herramientas de desarrollo del software

A continuación se describen las herramientas y tecnologías usadas en el desarrollo del software. En este caso para una mejor integración del componente con el proyecto se usa el lenguaje de alto nivel C++, el Entorno de Desarrollo Integrado Qt Creator, el marco de trabajo Qt y la biblioteca gráfica Qwt además de la metodología AUP tomada como estándar en la UCI. Se seleccionó además el lenguaje de modelado UML y la herramienta CASE Visual Paradigm para plasmar el ciclo vida del desarrollo del componente.

1.2.2 Visual Paradigm 5.0

Visual Paradigm para UML es una herramienta de modelado profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite crear diferentes tipos de diagramas en un ambiente totalmente visual. Es muy sencillo de usar, fácil de instalar y actualizar. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta de modelado también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos (ZIELIŃSKI, y otros,

2006)

1.2.1 Lenguaje Unificado de Modelado (UML).

El lenguaje de modelado utilizado fue el UML (Lenguaje de Modelado Unificado, por sus siglas en inglés), el cual es un lenguaje gráfico que permite visualizar, especificar, construir y documentar un sistema (Rodriguez, 2010). Es utilizado para especificar y describir métodos o procesos. Además, ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

1.2.3 Lenguaje de programación C++

Se utilizó C++ como lenguaje de programación para el desarrollo. C++ surge en 1980 gracias a Bjarne Stroustrup. Es un lenguaje imperativo orientado a objetos derivado del C. En realidad es un súper conjunto de C, que nació para añadirle cualidades y características de las que carecía, como son: las clases y el polimorfismo, los tipos de datos genéricos, la posibilidad de declarar variables en cualquier lugar del programa y además un motor de objetos con herencia múltiple, que permite combinar la programación imperativa de C, con la programación orientada a objetos. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción. (BJARNE, 1993)

1.2.4 Qt 4.8.6

Qt es un framework de desarrollo para aplicaciones multiplataforma que simplifica el desarrollo de aplicaciones en C++ de forma nativa. (Blanchette, y otros, 2008) Es una biblioteca que permite desarrollar interfaces gráficas de usuarios que además fue utilizado para desarrollar el HMI de Zamora R1 que es la versión del GALBA a la que tienen que integrarse el componente.

1.2.5 Qwt 6.1.0

La biblioteca de Qwt contiene componentes de GUI que son principalmente útil para los programas con una formación técnica. Al lado de una base para las tramas de 2D suministra balanzas, puertas corredizas, discos de marcar, compases, termómetros, ruedas y pomos para controlar o exhibir valores, selecciones, o alcances de tipo doble. (Qwt, 2015)

1.2.6 Qt Creator 2.5.0

Se utiliza para proveer un entorno de desarrollo integrado de plataforma múltiple y completo (IDE). A desarrolladores de aplicaciones les sirve para crear aplicaciones para plataformas de dispositivo de escritorio y movibles múltiples, como Android y iOS. Está disponible para Linux, X de OS de Mac y Windows. Cuenta con las siguientes características; editor de código con soporte para C++, herramientas para la rápida navegación del código, resaltado de sintaxis y auto-completado de código, control estático de código y estilo a medida que se escribe, ayuda sensitiva al contexto, plegado de código (code folding), paréntesis coincidentes y modos de selección.(Creator, 2014)

1.3 Metodologías de desarrollo de software

Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia (cumplir los requisitos iniciales) y la eficiencia (minimizar las pérdidas de tiempo) en el proceso de generación de software. (MOLPECERES) Las metodologías, en los últimos tiempos, se han dividido en dos grupos: los llamados métodos pesados y los métodos ligeros. La diferencia fundamental entre ambos es que mientras los métodos pesados intentan conseguir el objetivo común por medio de orden y documentación, los métodos ligeros (también llamados metodologías ágiles) tratan de mejorar la calidad del software por medio de una comunicación directa e inmediata entre las personas que intervienen en el proceso. A continuación la metodología de desarrollo de software seleccionada.

1.3.1 AUP

Para lograr la homogeneidad en el proceso de desarrollo de software, de los proyectos pertenecientes a los centros productivos de la UCI, y solucionar los errores encontrados, se decidió converger en una metodología que agrupe algunas de las características de las ya aplicadas, y sin alejarse de lo que hasta el momento se estaba poniendo en práctica, se propone el Proceso Unificado Ágil (AUP) de Scott Amber describe un método sencillo, fácil de entender para el desarrollo de software de aplicaciones empresariales utilizando técnicas ágiles y conceptos que aún permanecen todavía fieles a RUP. Esta metodología abarca siete flujos de trabajo, cuatro de ellos son de ingeniería y los otros tres de apoyo:

Modelado, implementación, prueba, despliegue, gestión de configuración, gestión de proyectos y ambiente. Dispone además de cuatro fases igual que RUP: incepción o creación, elaboración, construcción y transición. Este enfoque incluye diversas técnicas ágiles, desarrollo dirigido por pruebas, modelado ágil, gestión del cambio ágil y refactorización de base de datos para mejorar su productividad.

La metodología AUP se basa en los siguientes principios:

- Simplicidad: Todo se describe concisamente utilizando poca documentación, no miles de ellas.
- Agilidad: El ajuste a los valores y principios de La Alianza Ágil.
- Centrarse en actividades de alto valor: La atención se centra en las actividades que en realidad lo requieren, no en todo el proyecto.
- Herramienta de la independencia: Puede usar cualquier conjunto de herramientas que desea con el AUP. (PRIOLO, 2009)

1.4 Conclusiones parciales

Como resultado de la revisión de varios documentos y estudios teóricos relacionados con las cartas dinagráficas y los elementos cercanos al tema, se mostró la importancia y ventajas que tienen el desarrollo y utilización del componente. Por otro lado, se definieron y adoptaron las tecnologías, herramientas y la metodología de desarrollo a ser utilizadas en la implementación del componente para lograr un correcto funcionamiento del mismo.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

Introducción

El objetivo de este capítulo es iniciar el proceso de desarrollo del sistema propuesto como solución, así como identificar los principales requerimientos funcionales y no funcionales que debe tener el sistema para posteriormente detallar sus principales procesos, se realizará el modelo del dominio, se identificarán y se modelará el sistema completamente, todos estos con sus artefactos correspondientes.

2.1 Modelo de Dominio

Se considera de uno a más diagramas de clases UML que muestran los conceptos básicos del dominio del problema, sus propiedades y las relaciones más importantes entre dichos conceptos.

2.1.1 Diagrama de clases del dominio

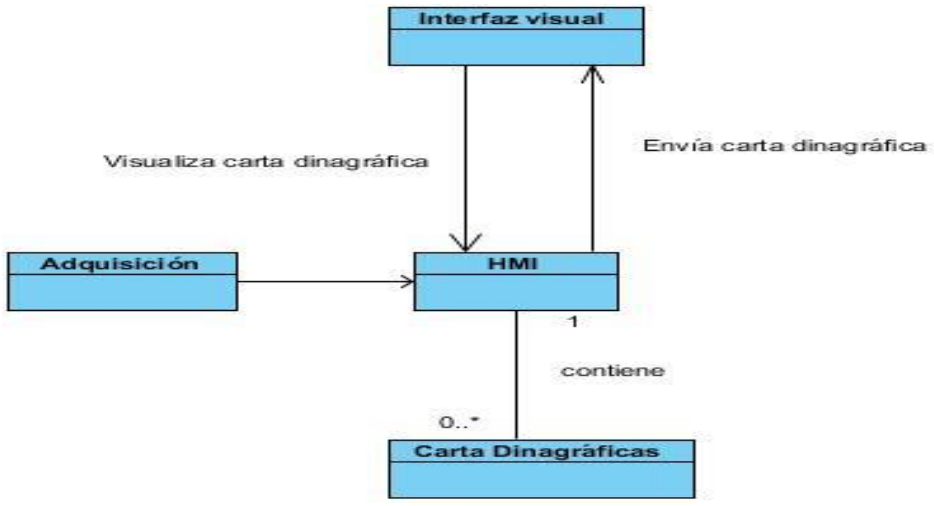


Ilustración 5.Diagrama de clases del dominio.

2.1.2 Descripción del modelo de dominio

Interfaz Visual: Es donde se visualiza las carta dinagráfica para el control de los procesos industriales.

HMI: Es el encargado de solicitar la carta dinagráfica.

Carta Dinagráfica: Contiene las cartas dinagráficas

Adquisición: Envía la carta dinagráfica

2.2 Descripción del sistema propuesto

Será un componente de interpretación y visualización de cartas dinagráficas, brindadas por el driver Modbus ELAM, el mismo va a ayudar a los ingenieros para la interpretación del estado de los pozos y alargará la vida de la cabilla de la bomba y de la bomba de extracción de petróleo, permitirá visualizar una o más cartas dinagráficas, visualizar la información de las mismas y así contribuir al correcto funcionamiento del SCADA GALBA.

2.2.1 Descripción de los actores

Tabla# 1 Actores del sistema.

Actor	Descripción
Mantenedor	Pueden ser un ingeniero del petróleo, geofísico u otra materia a fin, que tienen el conocimiento para editar las cartas, este puede desempeñar el rol de operador igualmente.

Operador	Un trabajador con un conocimiento básico que pueda lograr un criterio sobre el estado del pozo a partir de la carta.

2.3 Captura de requisitos

A continuación se muestran los requisitos funcionales y no funcionales. Los primeros son declaraciones de los servicios que debe proporcionar el componente, describe lo que este debe hacer. Los requisitos no funcionales son restricciones de los servicios, cualidades o propiedades que el producto debe tener. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, como cuán usable, seguro, conveniente o disponible puedan ser estos. (PRESSMAN, 2010-03-04)

2.3.1 Requisitos Funcionales

Tabla# 2 Requisitos funcionales.

Requisitos funcionales	Descripción
RF 1: Visualizar componente	Permite visualizar el componente al soltarlo sobre el despliegue.
RF 2: Mostrar en inspector de propiedades	Permite visualizar las propiedades a configurar para visualizar el componente en el despliegue.

RF 3: Adicionar carta dinagráfica	Permite adicionar una o varias cartas dinagráficas al componente.
RF 4: Eliminar carta dinagráfica	Permite eliminar una o varias cartas dinagráficas seleccionadas del componente.
RF 5: Mostrar información de la carta dinagráfica	Permite visualizar los datos de entrada que provienen del driver.
RF 6: Mostrar la leyenda	Permite mostrar la leyenda de las cartas dinagráficas previamente visualizadas.
RF 7: Configurar el área de pintado	Permite configurar el área de pintado del componente.
RF 8: Configurar rejillas	Permite configurar las rejillas del componente.
RF 9: Configurar eje X	Permite configurar el eje X del componente.
RF 10: Configurar eje Y	Permite configurar el eje Y del componente.
RF 11: Configurar propiedades generales	Permite configurar todas las propiedades comunes de los gráficos de HMI: importar XML, salvar XML, salvar JPG, configuración, acercar, alejar, restablecer tamaño, fondo blanco o negro del componente, mostrar u ocultar leyenda, imprimir, mostrar rejillas, restaurar por

	defecto.
RF 12: Configurar leyenda	Permite configurar la leyenda mostrada en el componente.
RF 13: Configurar gráfico de cartas	Permite configurar el componente gráfico.

2.3.2 Requisitos no Funcionales

- **Requisito de usabilidad**

RNF 1: El sistema debe ser usado por personal con conocimiento básico de informática.

RNF 2: El sistema debe ser usado sólo por personal capacitado en el SCADA GALBA.

RNF 3: Forma de interacción de la interfaz: Basada en ventanas y pestañas.

RNF 4: Interfaz de visualización de operaciones y sus atributos: La interfaz debe permitir efectuar todas las actividades de supervisión.

- **Requisito de Software**

RNF 5: Como Sistema Operativo GNU Linux Debian 7

- **Requisito de Hardware**

RNF 6: Un ordenador con al menos 512 MB de RAM, procesador Pentium/AMD 1.0 GHz o más.

- **Requisitos del diseño y la implementación**

RNF 7: Emplear el paradigma programación orientado a objetos.

RNF 8: Utilizar como lenguaje de programación C++.

- **Requisitos de Confiabilidad**

RNF 9: Debe mantenerse la consistencia de los datos en correspondencia con la realidad.

RNF 10: Debe existir una comunicación eficiente y fiable, evitando que no existan pérdidas de información de variables y eventos.

RNF 11: Integridad de las configuraciones o parámetros: El sistema debe ser capaz de mantener la integridad de las configuraciones o parámetros de las tareas, así como, los scripts asociados a ellas.

RNF 12: Manejo de datos en memoria: Ante una solicitud de fin de ejecución del sistema, se deben procesar todas las tareas que cumplieron la condición para su procesamiento.

- **Requisitos Legales**

RNF 13: Se utilizan herramientas de software libre de las cuales se posee licencia para su uso y distribución.

- **Requisitos de eficiencia**

RNF 14: Debe garantizar procesos y transacción masiva de datos: Hacia y desde todos los nodos y sistemas asociados, sin efectos negativos sobre el rendimiento del sistema.

RNF 15: Tiempo de respuesta en las consolas máximo de dos (2) segundos: Toda solicitud del operador debe presentar al menos una respuesta parcial en las consolas en un tiempo inferior a 2 segundos. En particular el tiempo para cambiar dos ventanas debe ser inferior a 1 segundo.

RNF 16: Tiempo de respuesta en las consolas máximo de dos (2) segundos: Toda solicitud del operador debe presentar al menos una respuesta parcial en las consolas en un tiempo inferior a 2 segundos. En particular el tiempo para cambiar dos ventanas debe ser inferior a 1 segundo.

RNF 17: Tiempo de respuesta para adquisición y procesamiento de los datos: (asociado a la velocidad de respuesta del dispositivo de control) 200 milisegundos, referido al periodo de encuesta mínimo o frecuencia máxima de muestreo.

RNF 18: Tiempo para el refrescamiento de datos en el componente gráfico de 1 - 2 segundos.

RNF 19: Tiempo de pintado: El sistema debe de contar con un tiempo de pintado de 2000 milisegundos.

RNF 20: Durante el intercambio de variables, debe existir una comunicación eficiente, garantizando una velocidad en la comunicación adecuada.

2.3.3 Historia de usuario

Las HU sustituyen a los documentos de especificación funcional, y a los “casos de uso”. Estas “historias” son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar (PÉREZ, 2012)). Las principales características de las historias de usuario son según (BECK, 1999) independientes unas de otras, negociables, valoradas por los clientes o usuarios, estimables, pequeñas y verificables.

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

- Número: Un número consecutivo, este permite identificar la historia.
- Usuario: Muestra el usuario que interactúa con la historia.
- Nombre: Nombre que identifica la HU.
- Prioridad del negocio: Esta característica es dada por el cliente con los valores: alta, media o baja en dependencia de la importancia y orden en que desean que sean implementadas.
- Riesgo de desarrollo: Muestra el riesgo que tendrá la historia al ser desarrollada.
- Programador responsable: Nombre del programador responsable de desarrollar la historia de usuario.

- Descripción: Breve descripción del proceso que define la historia.

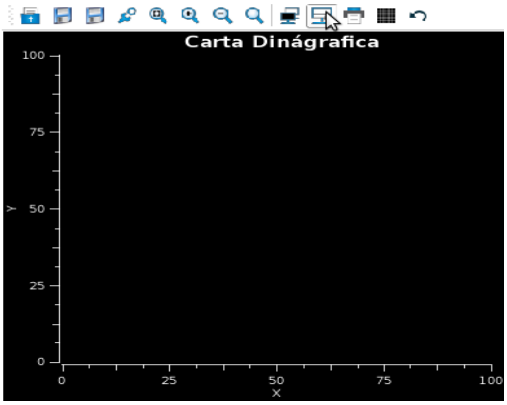
- Observaciones: Alguna acotación importante de señalar acerca de la historia.

- Prototipo de interfaz de usuario: Muestra cómo quedaran confeccionadas las interfaces de usuario.

A continuación se muestran las HU 1 Visualizar componente y HU 2 Mostrar en inspector de propiedades, los otros se encuentran en los Anexos.

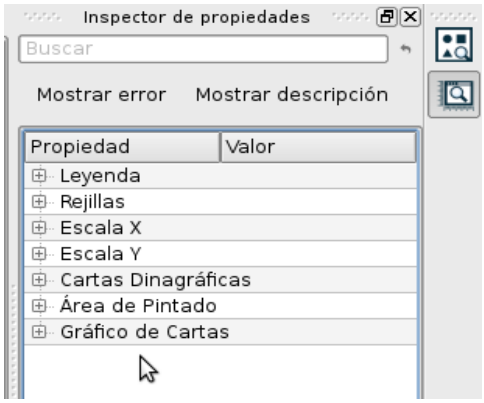
Tabla# 3 HU Visualizar componente.

Historia de Usuario	
Número: 1	**Usuario: Operador**
Nombre de Historia: Visualizar componente	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Programador responsable: Hansel Albellar Matos, Yurerlys Naranjo Palomino	
Descripción: El operador será capaz de visualizar el componente Carta Dinagráfica en un despliegue, para lograr esto el usuario selecciona el componente carta dinagráfica en la paleta de componente y lo suelta en un despliegue seleccionado previamente.	
Observaciones:	

• El componente solo de muestra arrastrándolo hacia un despliegue. • Debe de estar mostrado el componente para continuar trabajando en las siguientes historias de usuario.
Prototipo de interfaz de usuario: 

Tabla# 4 HU Mostrar inspector de propiedades.

Historia de Usuario	
Número: 2	**Usuario: Operador**
Nombre de Historia: Mostrar en inspector de propiedades	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta

Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino
Descripción: El operador será capaz de visualizar las propiedades del componente cartas dinagráficas, para su posterior edición.
Observaciones: • Debe de haberse mostrado el componente carta dinagráfica en un despliegue.
Prototipo de interfaz de usuario: 

2.4 Modelo de diseño

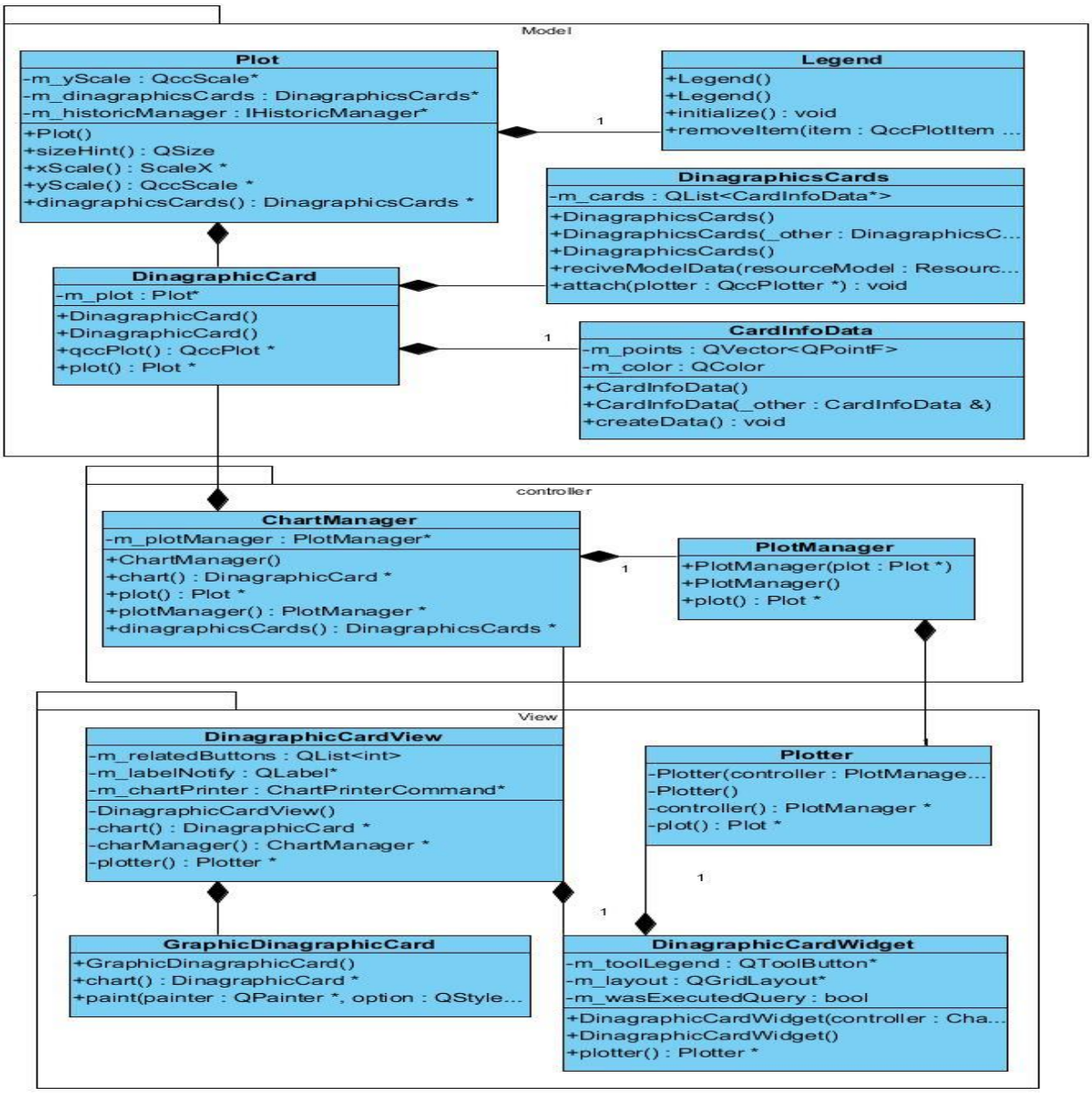
El diseño tiene el propósito de formular los modelos que se centran en los requisitos no funcionales y en el dominio de la solución y que prepara para la implementación y prueba del sistema. Pretende crear un plano

del Modelo de Implementación, por lo que el grueso del esfuerzo está en las últimas iteraciones de elaboración y las primeras de construcción.

2.4.1 Diagrama de clases de diseño

Los diagramas de clase de diseño representan las definiciones de las clases de software, mediante clases, con sus asociaciones y atributos, métodos, interfaces con sus operaciones y constantes, nos da la idea de cómo van a ser las dependencias y la navegabilidad en el sistema.

Ilustración 6 Diagrama de clases de diseño.



2.5 Arquitectura del Sistema

La arquitectura de software juega un papel fundamental en el diseño y la implementación de estructuras de software de alto nivel (Reyes, 2011). Es resultado de unir varios elementos arquitectónicos en función de satisfacer las funcionalidades y requerimientos de un sistema. Los patrones arquitectónicos constituyen un ejemplo de los elementos antes mencionados, proporcionando plantillas de trabajo que rigen el diseño

de la estructura general de un sistema (Pressman, 2005), permitiendo además que los miembros del equipo de desarrollo lo observen de una manera similar. El HMI propone una arquitectura ya definida para la visualización de los componentes gráficos tipo Sharp, la misma basada en el Modelo-Vista-Controlador el cual permite separar los datos de la aplicación (modelo), la interfaz de usuario (vista), y la lógica de control (controlador) en tres componentes distintos; donde el controlador funciona como intermediario entre el flujo de datos entre el modelo y la vista, como se ilustra a continuación, pero con especificidades para lograr efectos deseados por el sistema. Cualquier componente que se desee, debe heredar de estas clases ya definidas y reimplementar las funcionalidades, para lograr las particularidades del componente así como la forma de pintado.

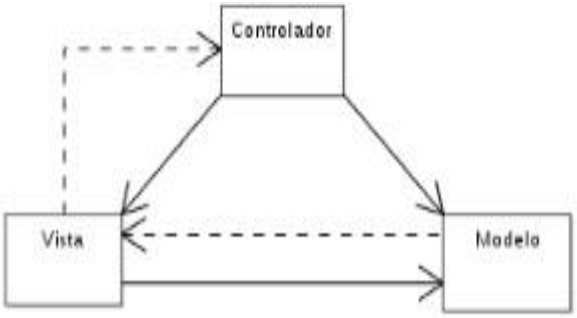


Ilustración 7.Patrón Modelo-Vista-Controlador

2.6 Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces (Vidal, 2011). Se decir que son soluciones a diferentes clases de problemas conocidos que pueden ser aplicadas a diferentes códigos (Delgado, y otros, 2012) Para realizar la solución, y garantizar una buena práctica de programación, se utilizaron varios patrones de diseño,

Para realizar la solución, y garantizar una buena práctica de programación, se utilizaron varios patrones de diseño, ejemplo el patrón observador, instancia única y fábrica abstracta.

2.6.1 Observador

El patrón observador en inglés observer que define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes. Se trata de un patrón de comportamiento, es decir, está relacionado con algoritmos de funcionamiento y asignación de responsabilidades a clases y objetos. (Timothy G. Mattson, 2008) Se utiliza en el momento de actualizar las mediciones representadas en el componente, inmediatamente se actualizan los observadores, siendo utilizado en la clase Plotter, observador de la clase Plot, y a su vez Plot sería observador de Legend y DinagraphicCard,

2.6.2 Instancia única

El patrón instancia única en inglés singleton se implementa creando en la clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado). Garantiza que una clase solo tenga una instancia y proporciona un punto de acceso global a ella. (E. Gamma, 1994)

2.6.3 Método plantilla

El patrón método plantilla en inglés template method define en una operación la estructura de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura, utilizado en el proceso de visualización de los objetos gráficos del SVG los cuales tienen operaciones comunes en los momentos iníciales solo diferenciándose en los finales de este proceso. (E. Gamma, 1994) Se usa para pintar las curvas, porque necesario utilizar el método de pintado ya existentes de la biblioteca *Qwt*.

2.6.4 Fábrica Abstracta

El patrón fábrica abstracta en inglés abstract factory hace visible el tipo de familia que se está usando, permitiendo trabajar con objetos de distintas familias sin mezclarlas. Este patrón permite crear distintas interfaces gráficas en el despliegue.

2.6.5 Patrones GRASP

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP) en inglés General Responsibility Assignments Software Patterns, describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. De acuerdo con el diseño se utilizaron los siguientes:

Patrón Experto: Asignan una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Nos indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. (Larman, 2003)

Patrón Creador: El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, o usa directamente las instancias creadas del objeto, o almacena o maneja varias instancias de la clase contiene o agrega la clase. (Larman, 2003)

2.7 Conclusiones parciales

Esta investigación conformó la propuesta de solución basándose en el análisis y estudio sobre la documentación existente referente a las tecnologías y herramientas actuales, así como un acercamiento a los principales conceptos, características y métodos a utilizar, se plasmaron las historias de usuario para la comprensión de lo que se quiere lograr y haciendo uso de la metodología AUP modelamos cual sería el resultado del componente a implementar.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Introducción

El presente capítulo describe la realización de la implementación del componente de acuerdo con el estilo de programación adoptado y las funciones y clases diseñadas previamente. Además se muestran los componentes utilizados en la realización del componente, los mecanismos para la creación de las cartas, las pruebas realizadas al mismo y los resultados obtenidos.

3.1 Estilos de programación

Los estilos de programación son convenios para escribir un código fuente y que después otros puedan entenderlo, de esta manera se logra que el mismo se pueda mantener. Para la implementación del componente se usó el lenguaje de programación C++ ya antes expuesto y su estilo de programación:

- **Licencia:** En los archivos cabecera debe incluir el copyright y la licencia, o una referencia de la misma, al estilo GNU GPL.
- **Plantilla para documentar clases, métodos, atributos y código en general**:

  Se utilizará la herramienta de auto-documentación *Doxygen* para generar la documentación del código de los distintos módulos involucrados en el proyecto y los formatos brindados por esta herramienta los cuales se explican a continuación:

  1. Se adopta el estilo de bloques de documentación *JavaDoc*, el cual consiste de un bloque de comentario de estilo C.

  2. Para hacer una descripción breve se adopta el uso del comando \brief o @brief en el bloque de comentarios ya descrito. La acción de este comando termina al final de un párrafo, de tal manera que la descripción detallada sigue después de una línea vacía.

  3. A la hora de hacer una descripción de los argumentos de métodos y funciones esta se hará en línea, es decir, luego de la declaración de cada uno de los argumentos se da una breve descripción de cada uno de ellos.

4. Para la documentación de tipos de datos definidos tales como enumerados, uniones o typedefs se pueden utilizar los formatos especificados en los apartados anteriores para describir su función y los elementos que los componen.

5. En algunos casos es necesario mostrar en la documentación una parte del código cuando se hacen comentarios entre líneas o para mostrar algún método en particular, ya que esto puede ahorrar hacer una explicación más extensa; para realizar esto en *Doxygen* se utiliza el comando ***@code*** y finaliza la acción del mismo con el comando ***@endcode.***

6. Es importante que se especifique el nombre del autor y la fecha de creación de cualquier estructura en un código, para ello se utilizan los comandos @author y @date para el nombre del autor y la fecha respectivamente.

7. Existe otro comando útil que permite hacer referencias a otras clases o métodos cuando esto sea necesario, es importante usar este comando en la documentación a la hora de implantar los métodos o funciones propias de alguna clase, este comando es *@see.*

8. El código será escrito en inglés y la documentación en español. Las variables y funciones comienzan con letra minúscula. Cada palabra consecutiva en el nombre comienza con letra mayúscula. (Lorenzo, 2010-2013)

3.2 Diagrama de componentes

Los diagramas de componentes muestran los componentes en los que está dividido un sistema y las dependencias entre ellos. UML define cinco estereotipos estándar que se pueden aplicar en los componentes los cuales son: ejecutable en inglés executable; componente que se puede ejecutar, biblioteca en inglés library; biblioteca de objetos estática o dinámica, tabla en inglés table; componentes que representa una tabla de base de datos, archivo en inglés file; componente que representa un documento que contiene código fuente o datos y por último documento en inglés document; componentes que representa un documento. A continuación se muestra el diagrama de componentes para la solución:

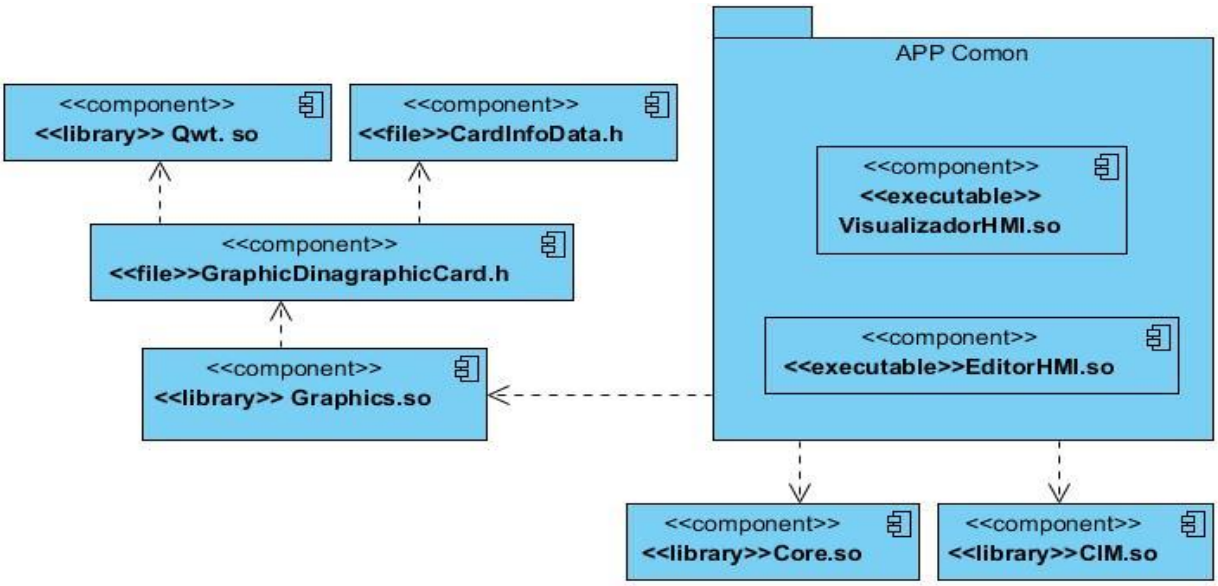


Ilustración 8.Diagrama de componentes.

El sistema general cuenta con elementos comunes para los ambientes de ejecución, el Editor HMI y Visualizador HMI. Estos elementos comunes se dividen en varias bibliotecas especializadas como Core, CIM y Graphics, en el presente desarrollo se agrega a la biblioteca Graphics el componente GraphicDinagraphicCard que cuenta con un CardInfoData para la gestión de las distintas cartas. De igual manera, GraphicDinagraphicCard incorpora la biblioteca Qwt para la gestión de pintado del componente gráfico a visualizar.

3.3 Diagrama de despliegue

Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema, muestra la arquitectura del sistema desde el punto de vista de distribución. A continuación se muestra el del componente a desarrollar:

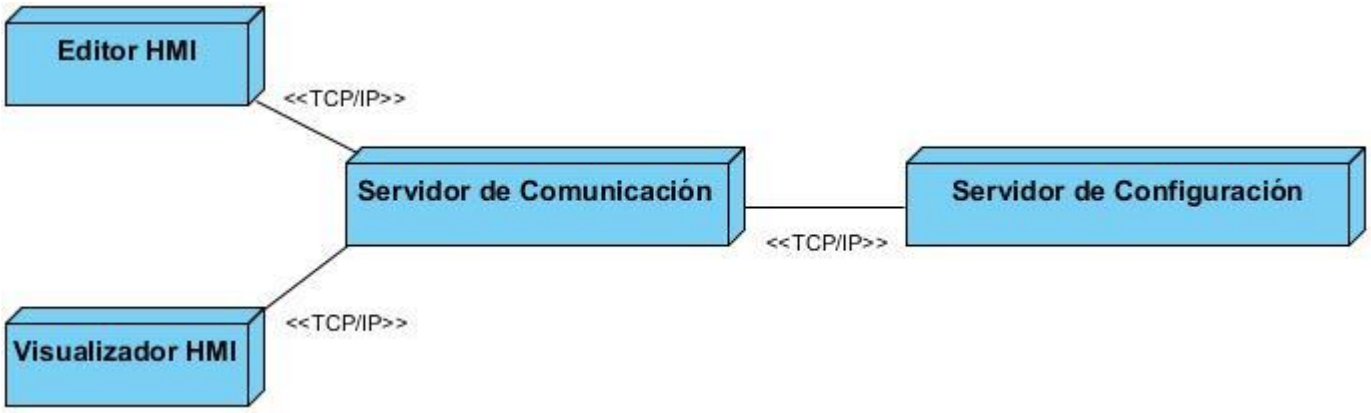


Ilustración 9.Diagrama de despliegue

3.4 Mecanismo para la representación gráfica del componente Cartas Dinagráficas

Cuando se selecciona la paleta que representa al componente Carta Dinagráfica dentro de la paleta de componentes para ser insertada en el despliegue, se invoca la clase *GraphicDinagraphicCard*, la misma se encarga de controlar las variaciones de la dimensión del mismo y recibir las actualizaciones en tiempo real a través del método *updateState*, es decir que se necesita una nueva jerarquía de clase que se encargue de la representación gráfica del componente, por lo que se propone la siguiente estructura.

El componente *GraphicDinagraphicCard* contiene una clase *DinagraphicCardView* que se encarga de la creación de las acciones sobre el área de pintado, dígase sea el caso de salvar una imagen en formato .JPG, mostrar el inspector de propiedades, opciones de agrandar o achicar la imagen o un área determinada, además restaura a tamaño normal, configurar la rejilla. También contiene un *DinagraphicCardWidget* que sería la encargada de la zona de pintado que se modificará según las

acciones del usuario sobre el componente como es la *Leyendas*. Para ello se crea la clase *Plotter* que es quien maneja dichas modificaciones.

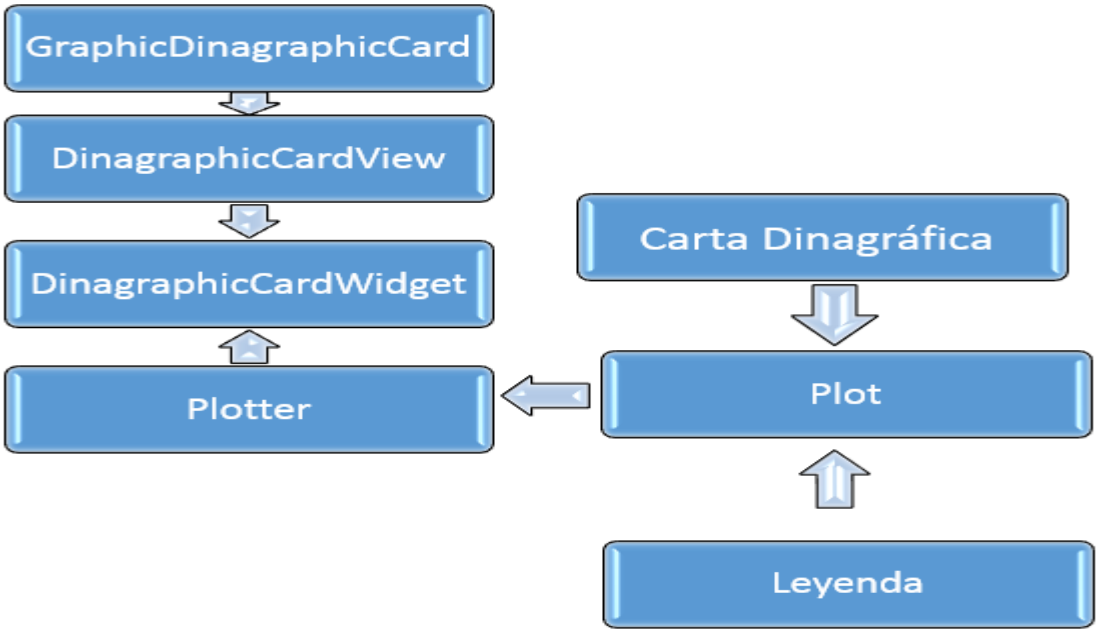


Ilustración 10.Prototipo de las clases principales

3.5 Lógica de creación de una carta dinagráfica

Paso 1: El usuario selecciona la opción Cartas Dinagráficas dentro del inspector de propiedades.

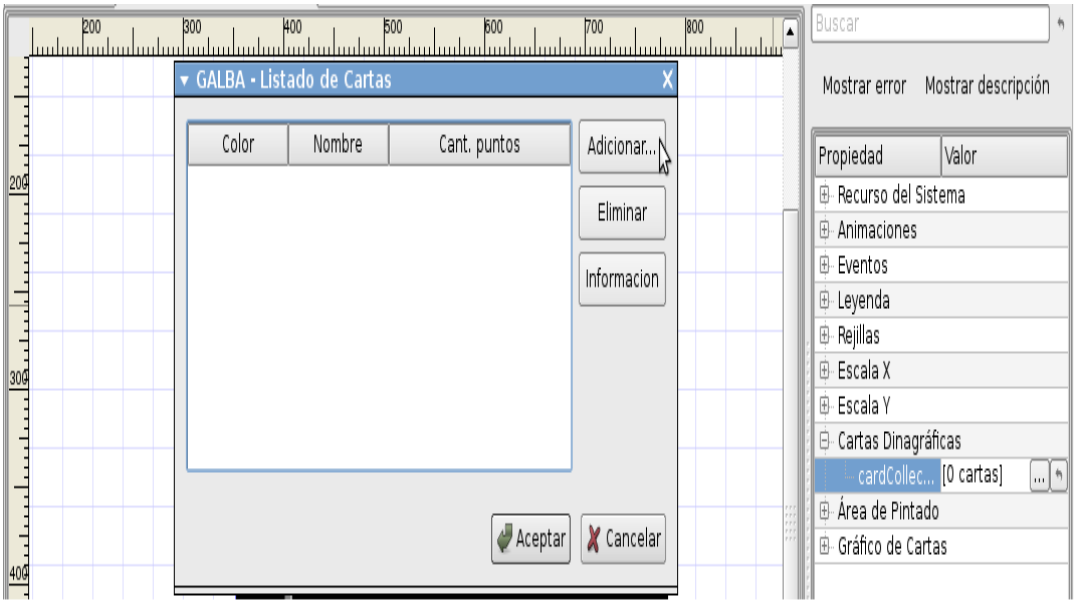


Ilustración 11 Paso 1 de la lógica de creación de una carta dinagráfica.

Paso 2: El usuario adiciona una a más Cartas Dinagráficas.

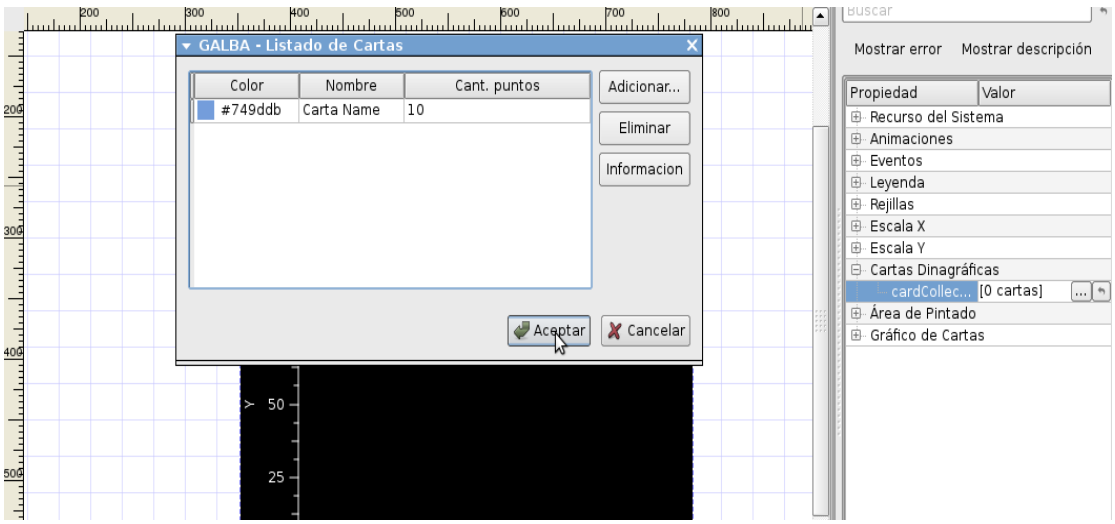


Ilustración 12.Paso 2 de la lógica de creación de una carta dinagráfica

Paso 3: (Opcional) El usuario muestra la información de la carta.

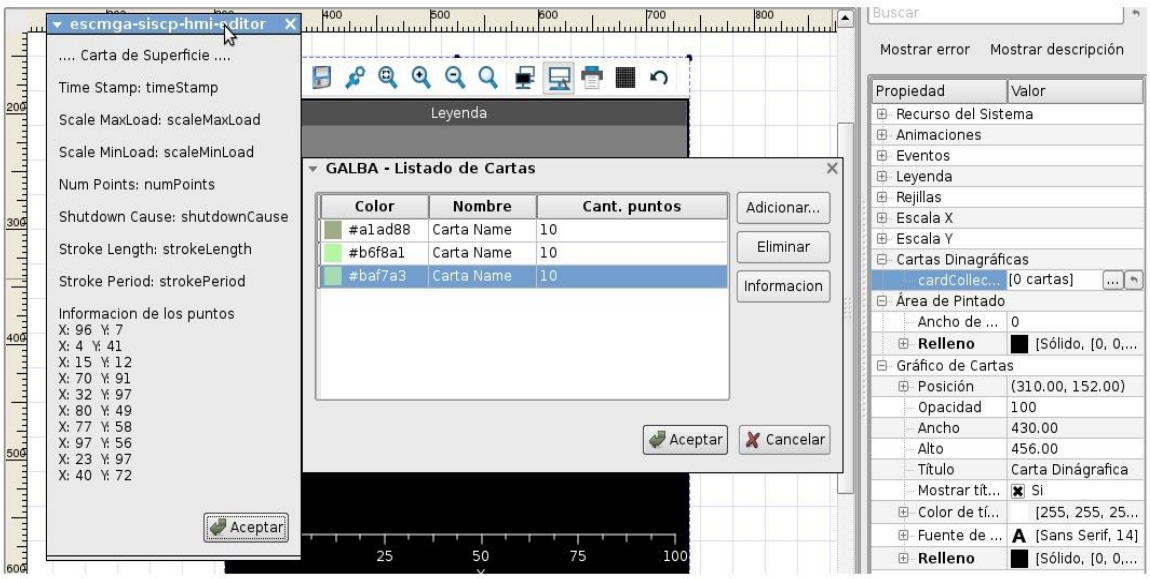


Ilustración 13.Paso 3 de la lógica de creación de una carta dinagráfica.

Paso 4: El usuario muestra una o más cartas.

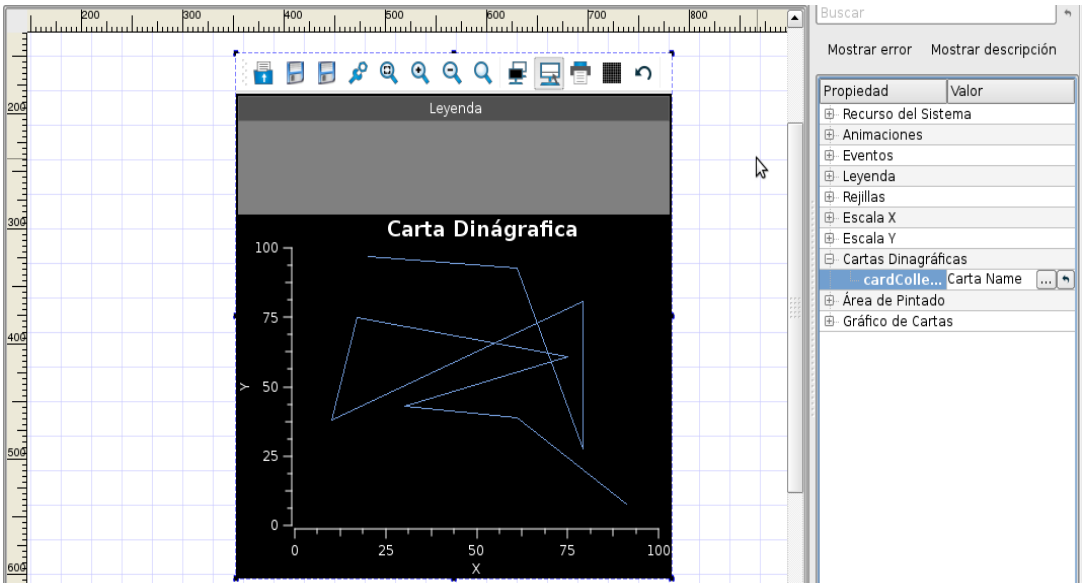


Ilustración 14.Paso 4 de la lógica de creación de una carta dinagráfica.

3.6 Tareas de ingeniería

Asociado a cada historia de usuario se encuentra la planificación de las tareas de ingeniería o programación, pues las mismas se transforman en estas tareas que son desarrolladas por programadores, dentro del equipo de desarrollo, aplicando la práctica de la programación en parejas. Se realizó la distribución de tareas en correspondencia con las HU que se desarrollaron.

Tabla# 5 Tareas de ingeniería por HU

No. HU	Tareas	No. Tarea
1	Creación del ícono de representación del componente en la paleta de componente.	1
	Seleccionar el ícono correspondiente en la paleta de componente.	2
2,3,4,5,6,7, 8,9,10,11, 12, 13	Crear propiedad para visualizar la ventana de configuración de las cartas dinagráficas dentro del inspector de propiedades.	3
	Asociar una carta dinagráfica a una curva.	4
2, 3, 4, 5, 6, 7	Crear la lógica de interpretación de la carta dinagráfica	5

Tabla# 6 Tarea de ingeniería # 1.

Tareas de ingeniería	
Número de la tarea: 1	**Número de la HU:** 1
Nombre de la tarea: Creación del ícono de representación del componente en la paleta de componente.	
Tipo de tarea: Desarrollo	**Días estimados:** 2
Programador(es) responsables: : Hansel Albellar Matos, Yurerlys Naranjo Palomino	
Descripción: Se implementa una lógica de incorporación de un nuevo IPalette dentro del módulo Extension.	

Tabla# 7 Tarea de ingeniería # 2.

Tareas de ingeniería	
Número de la tarea: 2	**Número de la HU:** 1
Nombre de la tarea: Seleccionar el ícono correspondiente en la paleta de componente.	
Tipo de tarea: Desarrollo	**Días estimados:** 3

Programador(es) responsables: : Hansel Albellar Matos, Yurerlys Naranjo Palomino
Descripción: Se implementa la lógica de utilización del patrón Abstract Factory para crear el componente gráfico deseado.

Tabla# 8 Tarea de ingeniería # 3.

Tareas de ingeniería	
Número de la tarea: 3	**Número de la HU:** 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
Nombre de la tarea: Crear propiedad para visualizar la ventana de configuración de las cartas dinagráficas dentro del inspector de propiedades.	
Tipo de tarea: Desarrollo	**Días estimados:** 3
Programador(es) responsables: : Hansel Albellar Matos, Yurerlys Naranjo Palomino	
Descripción: Implementar la lógica de creación de la propiedad DinagraphicCard dentro del módulo Property para reconocer la misma en el inspector de propiedades.	

Tabla# 9 Tarea de ingeniería # 4.

Tareas de ingeniería	
Número de la tarea: 4	**Número de la HU:** 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
Nombre de la tarea: Asociar una carta dinagráfica a una curva.	
Tipo de tarea: Desarrollo	**Días estimados:** 3
Programador(es) responsables: : Hansel Albellar Matos, Yurerlys Naranjo Palomino	
Descripción: Implementar la lógica de asociación de una carta dinagráfica con las curvas definidas de la biblioteca gráfica Qwt.	

Tabla# 10 Tarea de ingeniería # 5.

Tareas de ingeniería	
Número de la tarea: 5	**Número de la HU:** 2, 3, 4, 5, 6, 7

Nombre de la tarea: Crear la lógica de interpretación de la carta dinagráfica	
Tipo de tarea: Desarrollo	**Días estimados:** 4
Programador(es) responsables: : Hansel Albellar Matos, Yurerlys Naranjo Palomino	
Descripción: Implementar la clase que contiene la lógica de asociación de una carta dinagráfica con los datos extraídos por el driver Modbus ELAM.	

3.7 Pruebas del componente propuesto.

Probar es imprescindible para verificar la calidad y el adecuado funcionamiento de un software. La prueba es un proceso de ejecución de un programa con la intención de comprobar que el producto satisface los requerimientos y se comporta como se desea. Para que las pruebas tengan éxitos es necesario realizar casos de prueba que tengan probabilidad de descubrir los errores en el sistema y utilizar técnicas que nos guíen en el proceso de las pruebas. (Pressman, 2010-03-04) En este caso se hacen pruebas de aceptación o de caja negra.

3.7.1 Pruebas de caja negra

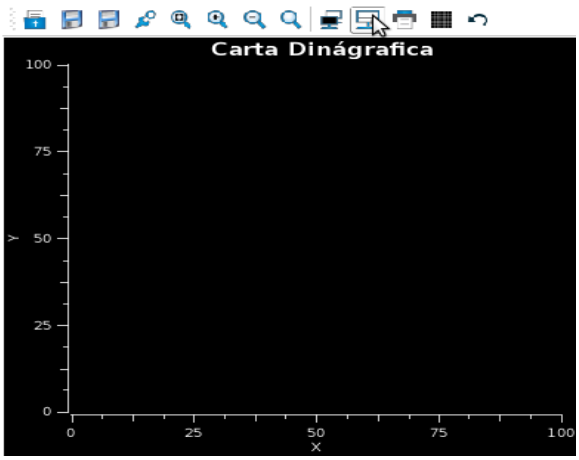
Las técnicas de caja negra o funcionales son las que se realizan sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica de los programas, únicamente la funcionalidad que debe realizar. También conocidas como pruebas de comportamiento, estas pruebas se basan en la especificación del programa o componente hacer probado para elaborar los casos de prueba. Para aplicar las pruebas al sistema se determinaron los casos de pruebas en un conjunto de entradas, condiciones de ejecución y resultados esperados, con tal de verificar el cumplimiento de un objetivo en particular. En las siguientes tablas se muestran el diseño del caso de

prueba Visualizar componente y Mostrar en inspector de propiedades los otros se pueden encontrar en los Anexos

Tabla# 11 Prueba de aceptación # 1.

Prueba de Aceptación	
Código: HU1_P1	**HU:** 1
Nombre: Visualizar componente	
Descripción: Visualizar un componente carta dinagráfica de la paleta de componentes en un despliegue.	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue.	
Entrada/Pasos de Ejecución: **Pasos:**	
Paso 1: El operador selecciona el ícono carta dinagráfica y lo suelta dentro del despliegue. Paso 2: Automáticamente se crea el componente dentro del despliegue.	
Resultado Esperado: El componente carta dinagráfica se muestra dentro del despliegue.	

Resultado Obtenido:



Evaluación de la Prueba: Satisfactoria.

Tabla# 12 Prueba de aceptación # 2.

Prueba de Aceptación	
Código: HU2_P2	**HU:** 2
Nombre: Mostrar en inspector de propiedades.	
Descripción: Permite visualizar las propiedades a configurar para visualizar el componente en el despliegue. .	

Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue. Se debe haber soltado el componente en un despliegue.

Entrada/Pasos de Ejecución:

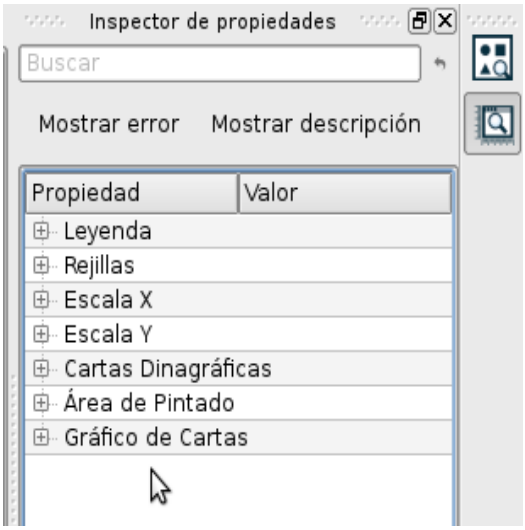
Pasos:

Paso 1:

Dar clic sobre el botón inspector de propiedades del sistema.

Resultado Esperado: El sistema muestra las propiedades del componente.

Resultado Obtenido:



Evaluación de la Prueba: Satisfactoria.

Resultados

Tabla# 13 Resultados de las pruebas de caja negra

Sistema	HU	Iteracción	NC	Cerrada	No Procede
Componente Carta Dinagraficás	13	1ra	30	28	2
		2da	10	10	0
		3ra	0	0	0

3.8 Conclusiones parciales

En la etapa de implementación de la solución propuesta se obtuvo la propuesta de solución dándole respuesta a todos los requisitos funcionales y no funcionales, para esto se adoptó el estilo de programación para la realización del SCADA GALBA, se realizaron pruebas de caja negra que legitiman los objetivos propuestos en las historias de usuario descritas, se desarrolló una serie de tareas de ingenierías argumentando el resultado alcanzado en la elaboración del componente cartas dinagráficas.

Conclusiones Generales

Conclusiones generales

Luego de la actualización del estudio del estado del arte, realizado la presente investigación sobre las cartas dinagráficas su uso y beneficios, y evidenciar la importancia del desarrollo del componente de interpretación y visualización de cartas dinagráficas para los pozos petroleros del SCADA GALBA, se hace posible arribar a las siguientes conclusiones:

- Con la realización del componente se logra una aplicación que interpreta y visualiza las cartas dinagráficas.

- Es posible lograr representaciones visuales con una mayor profesionalidad de representación gráfica, con la utilización de la librería Qwt, teniendo en cuenta que si se desea lograr una representación personalizada de visualización es necesario recurrir a la re implementación de la función de pintado.

- Las diferentes formas de configuración del componente propuesto hacen que el ambiente de trabajo sea lo más amigable posible para los operadores del sistema, logrando de esta forma la aceptación de los mismos.

- Es posible de conocer el estado estructural de los pozos petroleros, prevenir posibles incidentes indeseados asociados al derrumbe parcial o total del pozo, ahorrar inversión monetaria adicional en cabillas y contribuir a garantizar la calidad del producto extraído.

RECOMENDACIONES

Para darle continuidad a la investigación y el desarrollo los autores recomiendan:

- Incorporarle al componente la capacidad de comunicarse con el driver Modbus ELAM.
- Utilizar el componente en proyectos nacionales para que sus beneficios sean explotados.
- Lograr la integración con el sistema XDiag para Windows.

REFERENCIAS

2015. [En línea] 2015. http://iaci.unq.edu.ar/materias/laboratorio2/HMI\Introduccion HMI.pdf.

2015. [En línea] 2015. http://www.automatas.org/redes/scadas.htm.

Meza R, Edixon A y Pacheco R, Heisel Y. 2011. Trabajo especial de grado. *Diagnóstico en pozos con el sistema de producción de bombeo mecánico.* Venezuela : s.n., 2011.

BECK, K. 1999. *Extreme Programming Explained. .* 1999.

BJARNE, S. 1993. El lenguaje de programacion C++. s.l. : segunda ed. Addison-Wesley, 1993.

Blanchette, Jasmin y Summerfield, Mark. 2008. *C++ GUI Programming with Qt 4.* 2008.

Creator, Qt. 2014. Qt Creator Manual Digia Plc and/or its subsidiaries. 2014.

Delgado, Karel y Jiménez, Alejandro. 2012. *Módulo HMI para una tarjeta basada en microcontroladores.* La Habana : s.n., 2012.

E. Gamma, R. Helm, R. Johnson and J. Vlissides. 1994. *Design Patterns: elements of reusable object-oriented software.* s.l. : Addison-Wesley, 1994.

Larman, Craig. 2003. *UML y Patrones Introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Prentice Hall, 2003.

Ledesma Marcalla, Denis Alejandro y Hernández Cevallos, María Isabel. 2010. *Desarrollo de un sistema SCADA para la medición de voltajes con sistemas embebidos para el laboratorio de Mecatrónica de la Facultad de Mecánica.* 2010.

Lorenzo, Ariel Chávez. 2010-2013. *Estandares de codificaciones para C++.* 2010-2013.

MOLPECERES, A. java Hispano . [En línea]

http://www.javahispano.org/contenidos/es/procesos_de_desarrollo/

PÉREZ, J. 2012. *Guía Comparativa de Metodologías Ágiles. .* Segovia: Universidad de Valladolid. : s.n., 2012.

Pressman, R S. 2005. *Ingeniería del software. Un enfoque práctico.* 2005.

Pressman, R.S. 2010-03-04. *Ingenieria de Software.* s.l. : McGraw-Hill interamericana de España, 2010-03-04.

PRESSMAN, R.S. 2010-03-04. *INGENIERIA DE SOFTWARE.* s.l. : McGraw-Hill Interamericana de España S.L., 2010-03-04.

PRESSMAN, ROGER S. 2002. *Ingeniería del Software: Un Enfoque Práctico.* s.l. : McGraw-Hill Companies, 2002.

PRIOLO, S. 2009. *Métodos Agiles.* 2009.

Qwt. 2015. Qwt User´s Guide. [En línea] 2015. http://qwt.sourceforge.net/.

Reyes, Carlos Miguel Pérez. 2011. FISIM: SIMULADOR FÍSICO – MATEMÁTICO INTEGRADO A LA PLATAFORMA DE GESTIÓN DEL APRENDIZAJE ZERA. 2011.

Rodriguez, Pedro Uriarte. 2010. Manejador para la comunicacion con dispositivos de campo mediante el protocolo DF1. [En línea] 2010. http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_03836_10.

2007. Sistemas de Interfaz Hombre-Máquina,HMI. [En línea] noviembre de 2007. http://www.emb.cl/electroindustria/articulo.mvc?xid=837.

Thelin, Johan. 2007. Foundations of Qt Development. s.l. : Apress, 2007.

Timothy G. Mattson, Beverly A. Sanders, Berna L. Massingill. 2008. *Patterns for Parallel Programming.* s.l. : Addison- Wesley, 2008.

Vidal, Santiago. 2011. *IMPLEMENTACIÓN DEL SIMULADOR DEL TEMA DE GESTIÓN DE DISPOSITIVOS DE ENTRADA Y SALIDA PARA EL LABORATORIO VIRTUAL DE LA ASIGNATURA SISTEMAS OPERATIVOS. .* La Habana : s.n., 2011.

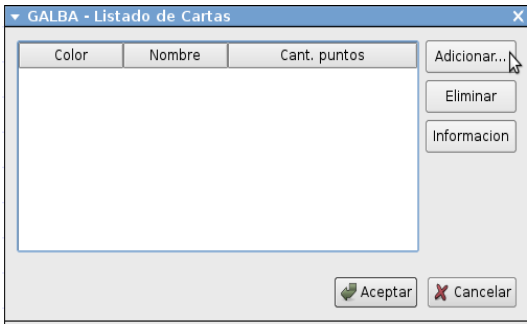
ZIELIŃSKI, K y SZMUC, T. 2006. *Software Engineering: Evolution And Emerging Technologies. .* s.l. : IOS Press, 2006. 448.

ANEXOS

Anexo # 1

Tabla# 14 HU Adicionar carta dinagráfica.

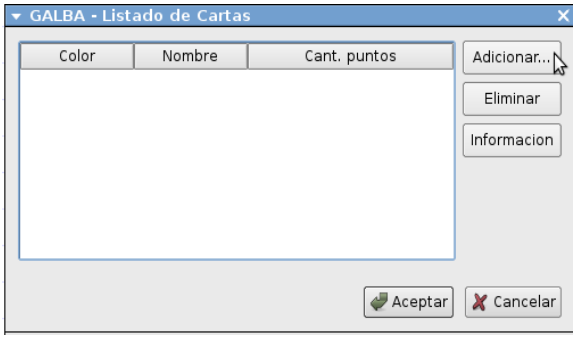
Historia de Usuario	
Número: 3	**Usuario: Operador**
Nombre de Historia: Adicionar carta dinagráfica.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino	
Descripción: El operador será capaz de agregar una carta dinagráfica en el componente ya insertado en el despliegue, para realizar esto, el operador accede a la propiedad Carta Dinagráfica del inspector de propiedades, luego da clic izquierdo en el botón cardColleccion, aparecerá otra ventana, que mostrará la opción de adicionar una carta dinagráfica, dando un clic izquierdo esta se adicionará con un color para su posterior identificación.	
Observaciones: • Las cartas dinagráficas adicionadas deben de mantener correspondencia con las	

extraídas por el driver.
Prototipo de interfaz de usuario: 

Anexo # 2

Tabla# 15 HU Eliminar carta dinagráfica.

Historia de Usuario	
Número: 4	**Usuario: Operador**
Nombre de Historia: Eliminar carta dinagráfica.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino	
Descripción: El operador será capaz de eliminar una carta dinagráfica en el componente ya insertado en el despliegue, para realizar esto, el operador accede a la propiedad Carta Dinagráfica del inspector de propiedades, luego da clic izquierdo en el botón cardColleccion,	

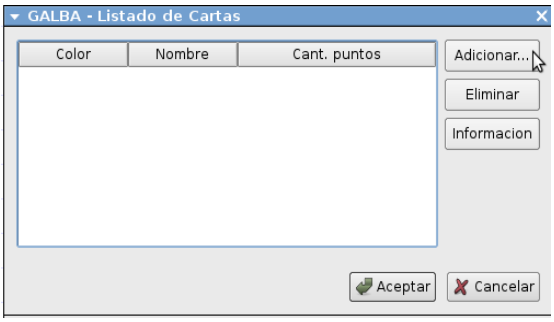
aparecerá otra ventana, que mostrará la opción de eliminar una carta dinagráfica, que elimina dando un clic izquierdo.
Observaciones: • Debe haber adicionada mínimo una carta.
Prototipo de interfaz de usuario: 

Anexo # 3

Tabla# 16 HU Mostrar información de la carta dinagráfica.

Historia de Usuario	
Número: 5	**Usuario: Operador**
Nombre de Historia: Mostrar información de la carta dinagráfica.	

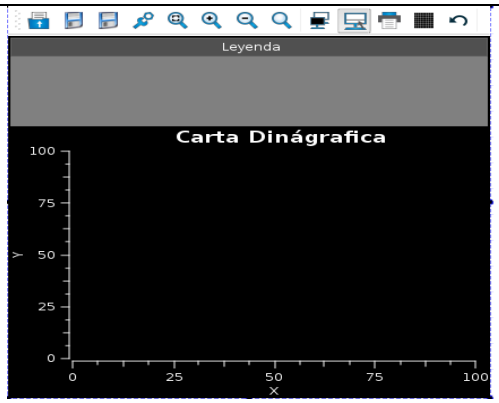
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino	
Descripción: El operador será capaz de mostrar la información de una carta dinagráfica en el componente ya insertado en el despliegue, para realizar esto, el operador accede a la propiedad Carta Dinagráfica del inspector de propiedades, luego da clic izquierdo en el botón cardColleccion, aparecerá otra ventana, que mostrará la opción de información de una carta dinagráfica, que dando un clic izquierdo mostrará la información.	
Observaciones: • Debe haber adicionada mínimo una carta dinagráfica para que se pueda mostrar la información de la misma.	
Prototipo de interfaz de usuario:	



Anexo #4

Tabla# 17 HU Mostrar la leyenda.

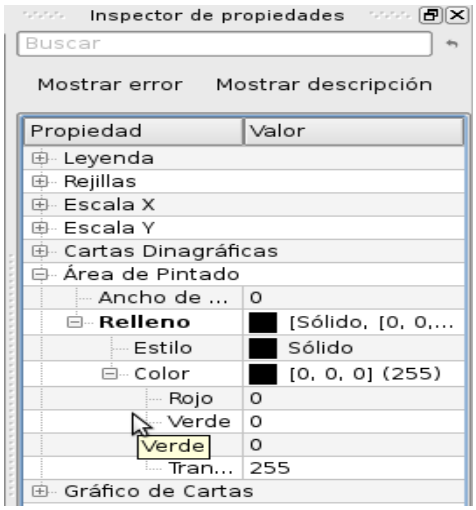
Historia de Usuario	
Número: 6	**Usuario: Operador**
Nombre de Historia: Mostrar la leyenda.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino	
Descripción: El operador será capaz de visualizar la leyenda de las cartas dinagráficas adicionadas previamente en el componente.	
Observaciones: • Debe haber adicionada mínimo una carta dinagráfica para que aparezca alguna en la leyenda.	
Prototipo de interfaz de usuario:	



Anexo # 5

Tabla# 18 HU Configurar el área de pintado.

Historia de Usuario	
Número: 7	**Usuario: Operador**
Nombre de Historia: Configurar el área de pintado.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino	
Descripción: El operador será capaz de editar el área de pintado del componente dependiendo de sus necesidades o especificaciones. Para esto debe acceder en el Inspector de Propiedades a la propiedad correspondiente a dicha área, este presenta la posibilidad de editar el ancho y el relleno.	

Observaciones: • Debe de haberse mostrado el componente carta dinagráfica en un despliegue.
Prototipo de interfaz de usuario: 

Anexo # 6

Tabla# 19 HU Configurar rejillas.

Historia de Usuario	
Número: 8	**Usuario: Operador**
Nombre de Historia: Configurar rejillas.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta

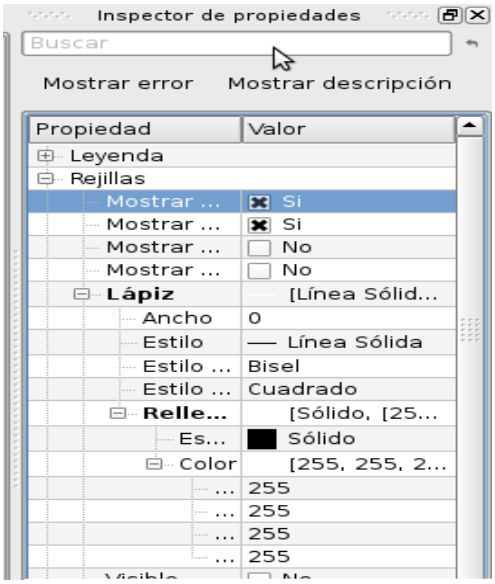
Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino

Descripción: El operador será capaz de editar la forma de visualización de las rejillas del componente. Para esto debe acceder en el Inspector de Propiedades a la propiedad correspondiente a las rejillas, en esta se brindan las opciones de si se desea o no mostrar dimensiones y las propiedades del lápiz con que se dibujarán las líneas.

Observaciones:

- Debe de haberse mostrado el componente carta dinagráfica en un despliegue.

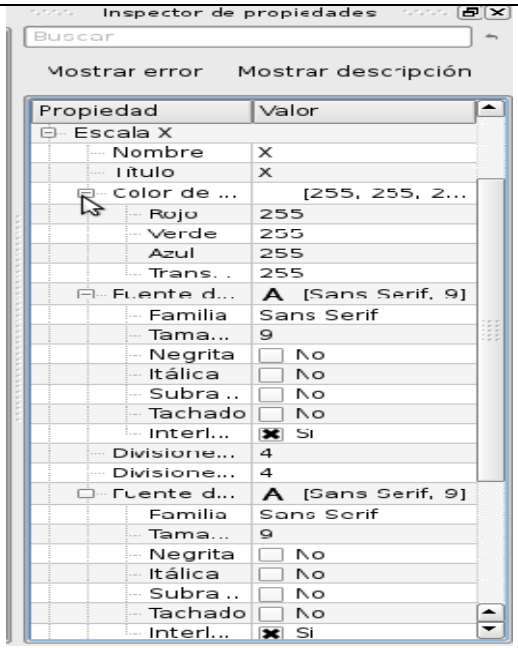
Prototipo de interfaz de usuario:



Anexo #7

Tabla# 20 HU Configurar eje X.

Historia de Usuario	
Número: 9	**Usuario: Operador**
Nombre de Historia: Configurar eje X.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino	
Descripción: El operador será capaz de editar la forma de visualización del eje X del componente. Para esto se accede dentro del inspector de propiedades a la propiedad correspondiente a editar dicho eje, esta da la posibilidad de editar el nombre, título del eje al igual que el color de la letra, la fuente y la forma que se pintará el eje.	
Observaciones: • Debe de haberse mostrado el componente carta dinagráfica en un despliegue.	
Prototipo de interfaz de usuario:	



Anexo # 8

Tabla# 21 HU Configurar eje Y.

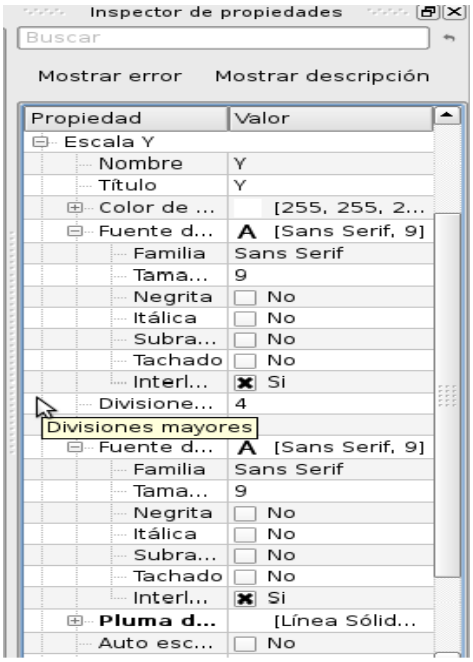
Historia de Usuario	
Número: 10	**Usuario: Operador**
Nombre de Historia: Configurar eje Y.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino	

Descripción: El operador será capaz de editar la forma de visualización del eje Y del componente. Para esto se accede dentro del inspector de propiedades a la propiedad correspondiente a editar dicho eje, esta da la posibilidad de editar el nombre, título del eje al igual que el color de la letra, la fuente y la forma que se pintará el eje.

Observaciones:

- Debe de haberse mostrado el componente carta dinagráfica en un despliegue.

Prototipo de interfaz de usuario:



Anexo # 9

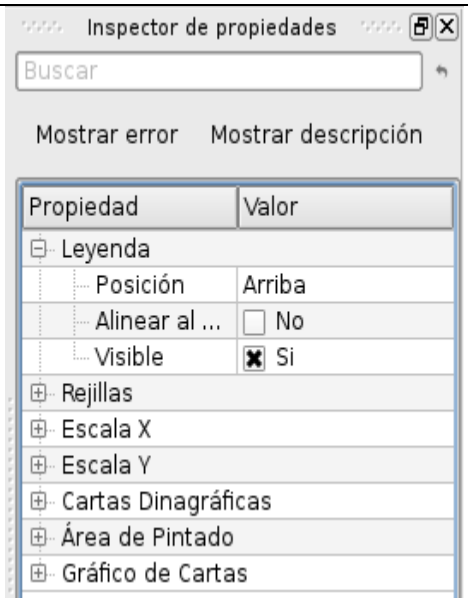
Tabla# 22 HU Configurar propiedades generales.

Historia de Usuario	
Número: 11	**Usuario: Operador**
Nombre de Historia: Configurar propiedades generales.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino	
Descripción: El operador será capaz de configurar propiedades generales del componente dígase sea el caso: importar XML, salvar XML, salvar JPG, configuración, acercar, alejar, restablecer tamaño, fondo blanco o negro del componente, mostrar u ocultar leyenda, imprimir, mostrar rejillas, restaurar por defecto.	
Observaciones: • Debe de haberse mostrado el componente carta dinagráfica en un despliegue. • Todas estas propiedades se encuentran en la parte superior del componente carta dinagráfica insertado en el despliegue.	
Prototipo de interfaz de usuario: 	

Anexo # 10

Tabla# 23 HU Configurar leyenda.

Historia de Usuario	
Número: 12	**Usuario: Operador**
Nombre de Historia: Configurar leyenda.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino	
Descripción: El operador será capaz de configurar las propiedades de la leyenda.	
Observaciones: • Debe de haberse mostrado el componente carta dinagráfica en un despliegue.	
Prototipo de interfaz de usuario:	



Anexo # 11

Tabla# 24 HU Configurar gráfico de cartas

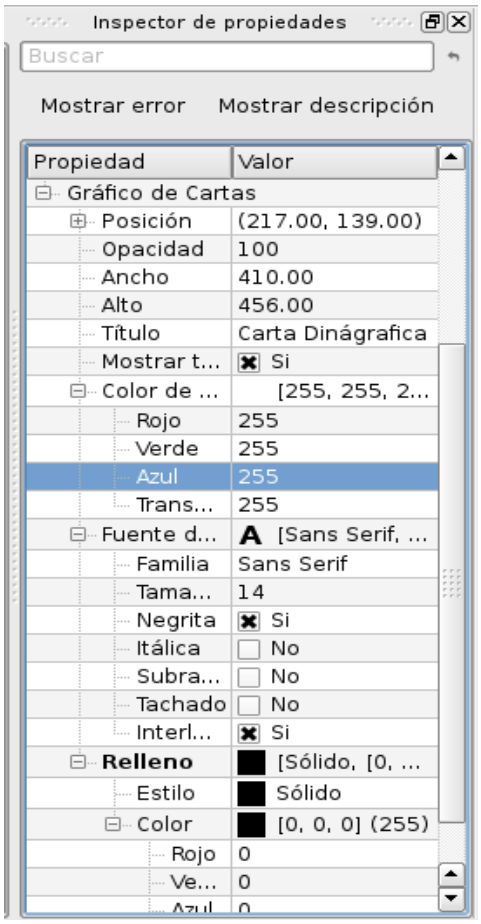
Historia de Usuario	
Número: 13	**Usuario: Operador**
Nombre de Historia: Configurar gráfico de cartas	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Programador responsable: Hansel Albellar Matos, Yurelys Naranjo Palomino	

Descripción: El operador será capaz de configurar las propiedades del componente gráfico.

Observaciones:

- Debe de haberse mostrado el componente carta dinagráfica en un despliegue.

Prototipo de interfaz de usuario:

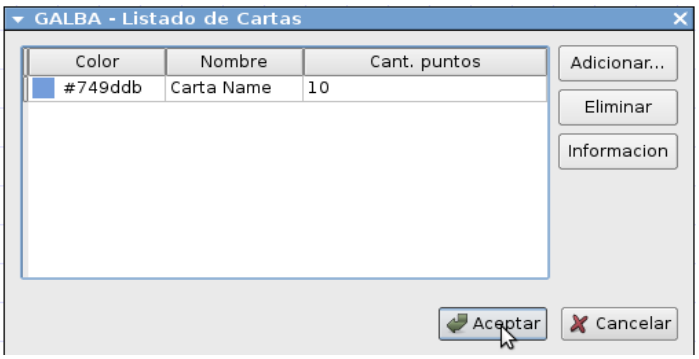


Anexo # 12

Tabla# 25 Prueba de aceptación # 3.

Prueba de Aceptación	
Código: HU3_P3	**HU:** 3
Nombre: Adicionar carta dinagráfica.	
Descripción: Permite adicionar una carta dinagráfica al componente.	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue. Se debe haber soltado el componente en un despliegue.	
Entrada/Pasos de Ejecución: **Pasos:**	
Paso 1: Selecciona la opción carta dinagráfica del inspector de propiedades. Paso 2: Clic en la opción cardColleccion Paso 3: En la nueva ventana dar clic en el botón adicionar	
Resultado Esperado: El componente adiciona una o más cartas.	

Resultado Obtenido:



Evaluación de la Prueba: Satisfactoria.

Anexo # 13

Tabla# 26 Prueba de aceptación # 4.

Prueba de Aceptación	
Código: HU4_P4	**HU:** 4
Nombre: Eliminar carta dinagráfica.	
Descripción: Permite eliminar una carta dinagráfica al componente.	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue. Se debe haber soltado el componente en un despliegue. Se debe haber adicionado una carta dinagráfica.	

Entrada/Pasos de Ejecución:

Pasos:

Paso 1:

Selecciona la opción carta dinagráfica del inspector de propiedades.

Paso 2:

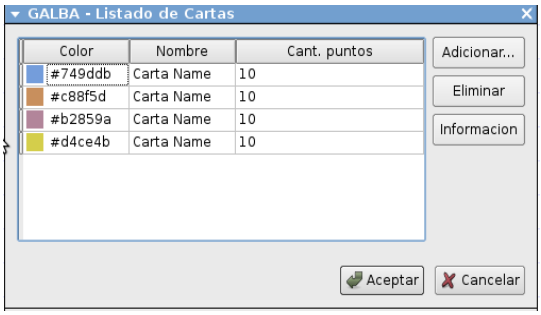
Clic en la opción cardColleccion

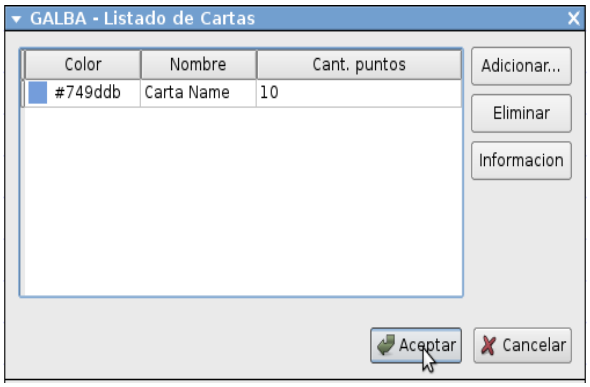
Paso 3:

En la nueva ventana dar clic en el botón eliminar

Resultado Esperado: El componente elimina una o más cartas.

Resultado Obtenido:





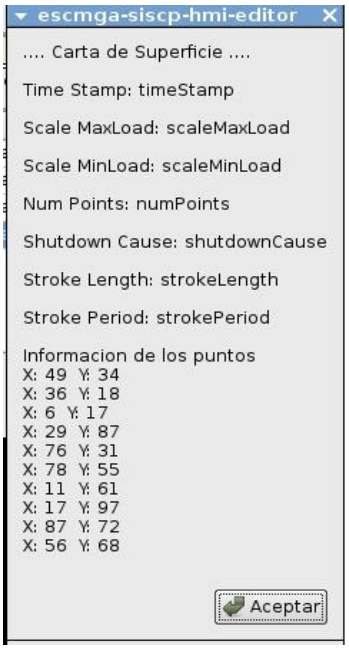
Evaluación de la Prueba: Satisfactoria.

Anexo # 14

Tabla# 27 Prueba de aceptación # 5.

Prueba de Aceptación	
Código: HU5_P5	**HU:** 5
Nombre: Mostrar información de la carta dinagráfica.	
Descripción: Permite visualizar la información de las cartas dinagráficas.	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue. Se debe haber soltado el componente en un despliegue. Se debe haber adicionado una carta dinagráfica.	

Entrada/Pasos de Ejecución: **Pasos:**
Paso 1: Selecciona la opción carta dinagráfica del inspector de propiedades. Paso 2: Clic en la opción cardColleccion Paso 3: En la nueva ventana dar clic en el botón información.
Resultado Esperado: El componente muestra la información de una carta.
Resultado Obtenido:



Evaluación de la Prueba: Satisfactoria.

Anexo # 15

Tabla# 28 Prueba de aceptación # 6.

Prueba de Aceptación	
Código: HU6_P6	**HU:** 6
Nombre: Mostrar la leyenda.	
Descripción: Permite visualizar la leyenda la des cartas dinagráficas. .	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener	

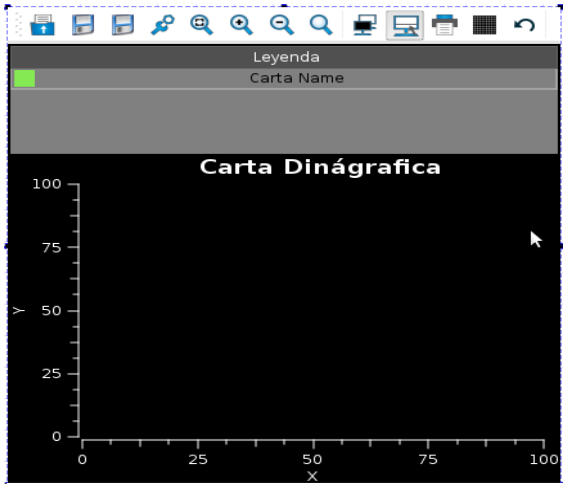
abierto un despliegue. Se debe haber soltado el componente en un despliegue. Se debe haber adicionado una carta dinagráfica.

Entrada/Pasos de Ejecución:

Pasos:

Resultado Esperado: El componente muestra la relación de las cartas adicionadas.

Resultado Obtenido:

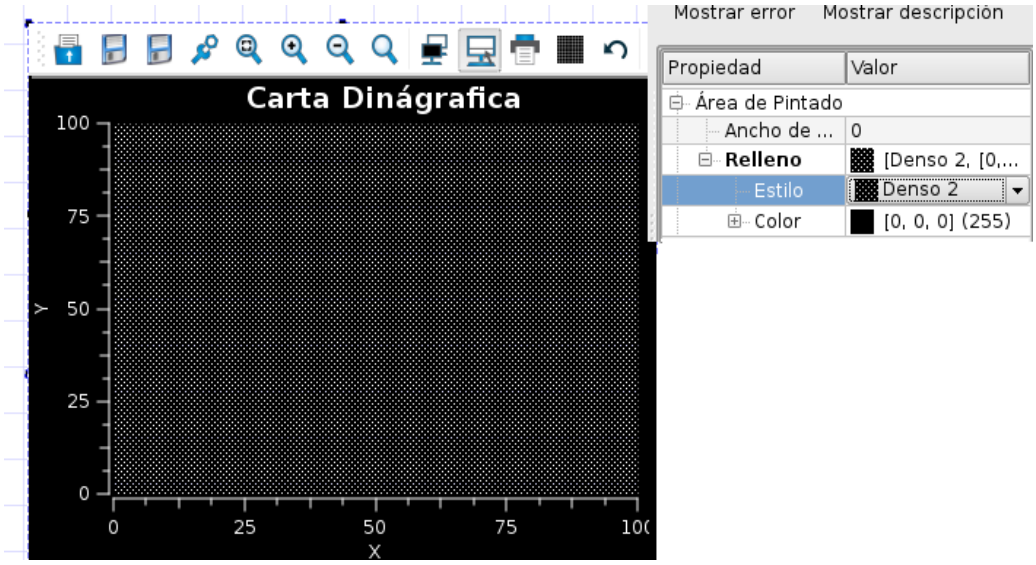


Evaluación de la Prueba: Satisfactoria.

Anexo # 16

Tabla# 29 Prueba de aceptación # 7.

Prueba de Aceptación	
Código: HU7_P7	**HU:** 7
Nombre: Configurar el área de pintado.	
Descripción: Permite configurar el área de pintado del componente cartas dinagráficas.	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue. Se debe haber soltado el componente en un despliegue.	
Entrada/Pasos de Ejecución: **Pasos:**	
Paso 1: Selecciona la propiedad área de pintado del inspector de propiedades. Paso 2: Modifica la opción que desee.	
Resultado Esperado: El componente modifica el área de pintado.	
Resultado Obtenido:	



Evaluación de la Prueba: Satisfactoria.

Anexo # 17

Tabla# 30 Prueba de aceptación # 8.

Prueba de Aceptación	
Código: HU8_P8	**HU:** 8
Nombre: Configurar rejillas.	
Descripción: Permite configurar los rejillas del componente.	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue. Se debe haber soltado el componente en un despliegue.	

Entrada/Pasos de Ejecución:

Pasos:

Paso 1:

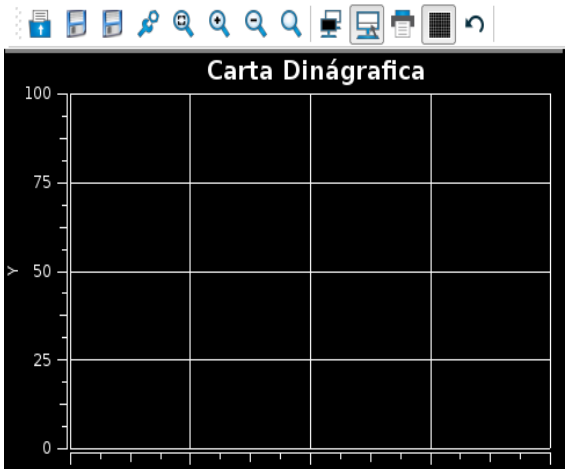
Selecciona la opción rejillas del inspector de propiedades.

Paso 2:

Modifica la opción que desee.

Resultado Esperado: El componente modifica la rejilla.

Resultado Obtenido:

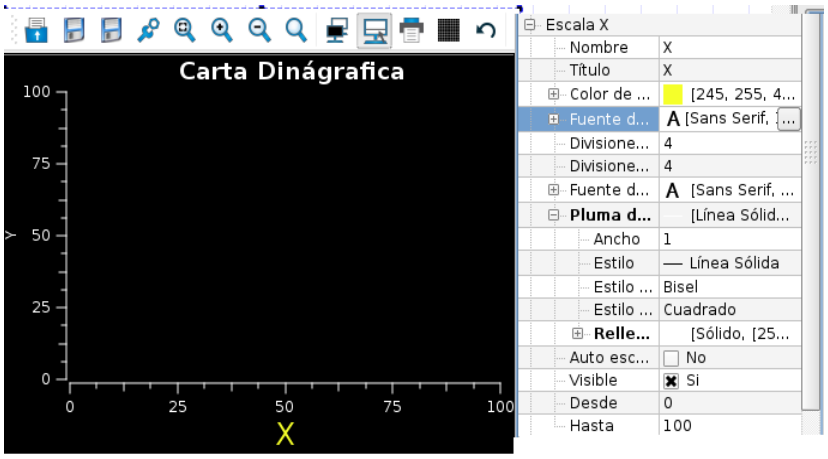


Evaluación de la Prueba: Satisfactoria.

Anexo # 18

Tabla# 31 Prueba de aceptación # 9.

Prueba de Aceptación	
Código: HU9_P9	**HU:** 9
Nombre: Configurar eje X	
Descripción: Permite configurar el eje X del componente.	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue. Se debe haber soltado el componente en un despliegue.	
Entrada/Pasos de Ejecución: **Pasos:**	
Paso 1: Selecciona la opción eje X del inspector de propiedades. Paso 2: Modifica la opción que desee.	
Resultado Esperado: El componente modifica el eje X.	
Resultado Obtenido:	

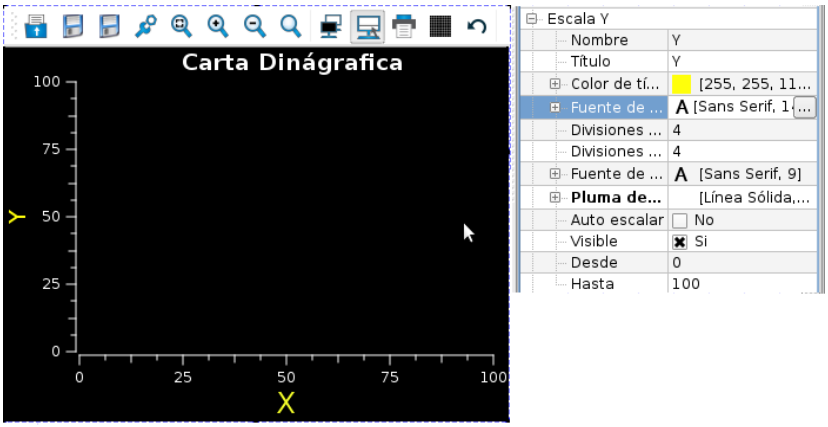


Evaluación de la Prueba: Satisfactoria.

Anexo # 19

Tabla# 32 Prueba de aceptación # 10.

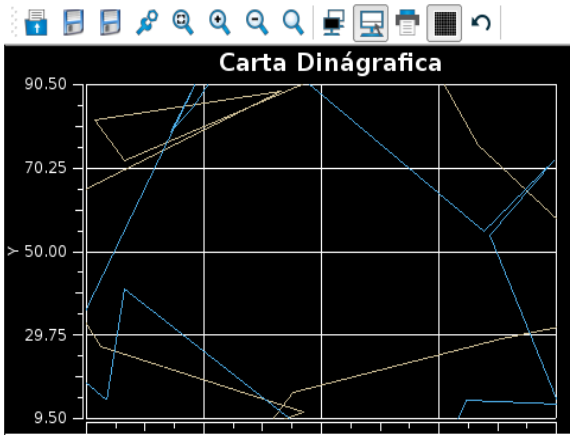
Prueba de Aceptación	
Código: HU10_P10	**HU:** 10
Nombre: Configurar eje Y	
Descripción: Permite configurar el eje Y del componente.	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue. Se debe haber soltado el componente en un despliegue.	
Entrada/Pasos de Ejecución: **Pasos:**	

Paso 1: Selecciona la opción eje Y del inspector de propiedades. Paso 2: Modifica la opción que desee.
Resultado Esperado: El componente modifica el eje Y.
Resultado Obtenido: 
Evaluación de la Prueba: Satisfactoria.

Anexo # 19

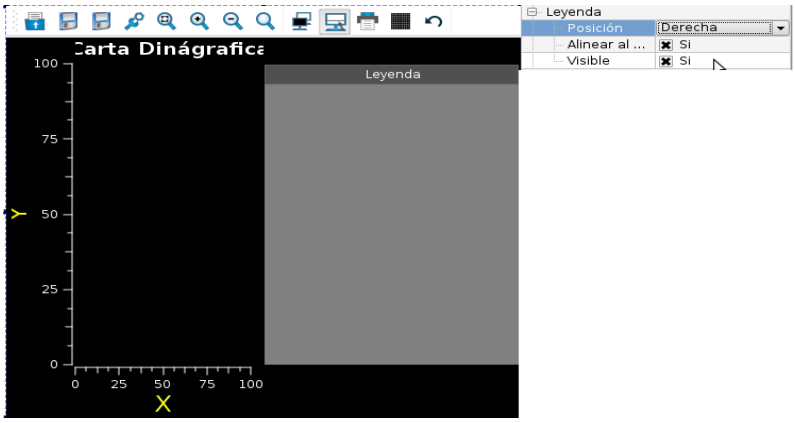
Tabla# 33 Prueba de aceptación # 11.

Prueba de Aceptación

Código: HU11_P11	**HU:** 11
Nombre: Configurar propiedades generales	
Descripción: Permite configurar propiedades generales.	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue. Se debe haber soltado el componente en un despliegue.	
Entrada/Pasos de Ejecución: **Pasos:**	
Resultado Esperado: El componente modifica cualquier propiedad común, en este caso aumentar las cartas y poner la rejilla.	
Resultado Obtenido: 	
Evaluación de la Prueba: Satisfactoria.	

Anexo # 20

Tabla# 34 Prueba de aceptación # 12.

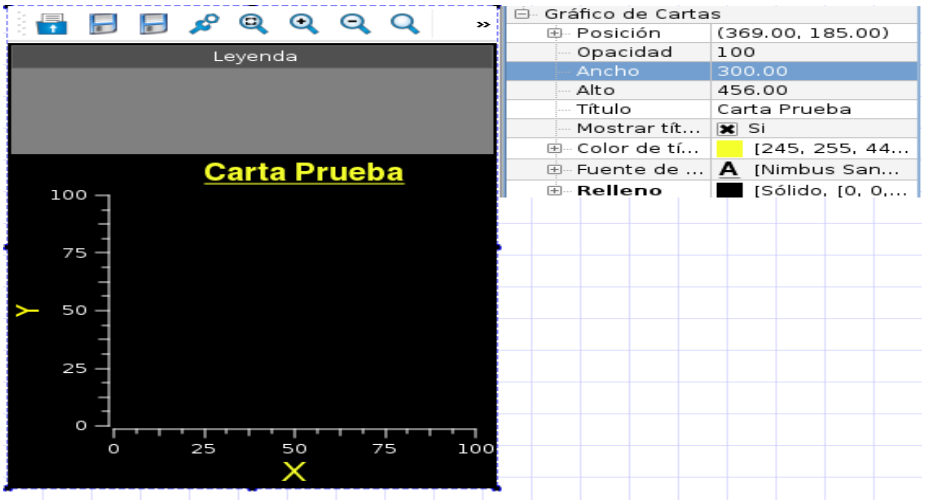
Prueba de Aceptación	
Código: HU12_P12	**HU:** 12
Nombre: Configurar leyenda	
Descripción: Permite configurar las propiedades de la leyenda	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue. Se debe haber soltado el componente en un despliegue.	
Entrada/Pasos de Ejecución: **Pasos:**	
Resultado Esperado: El componente modifica la leyenda.	
Resultado Obtenido: 	

Evaluación de la Prueba: Satisfactoria.

Anexo # 21

Tabla# 35 Pruebas de aceptación # 13.

Prueba de Aceptación	
Código: HU13_P13	**HU:** 13
Nombre: Configurar gráfico de cartas.	
Descripción: Permite configurar las propiedades del componente gráfico.	
Condiciones de Ejecución: Debe estar inicializado el sistema. El usuario debe de tener abierto un despliegue. Se debe haber soltado el componente en un despliegue.	
Entrada/Pasos de Ejecución: **Pasos:**	
Resultado Esperado: El componente se modifica.	
Resultado Obtenido:	



Evaluación de la Prueba: Satisfactoria.