Universidad de las Ciencias Informáticas
FACULTAD 6

Título: Automatización de Acciones de Usuarios para GeoQ.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Ariel Triana Rodríguez

Tutor: MSc. David Silva Barrera

Junio, 2012

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Ariel Triana Rodríguez

_____
Firma del Autor

David Silva Barrera

_____
Firma del Tutor

DATOS DE CONTACTO

Autor:

Nombre: Ariel.

Apellidos: Triana Rodríguez.

Correo electrónico: atrodriguez@estudiantes.uci.cu

Tutor:

Nombre: David.

Apellidos: Silva Barrera.

Correo electrónico: dsilva@uci.cu

Categoría Docente: MSc.

AGRADECIMIENTOS

A mi Mamá, porque es lo que más quiero en la vida y por comprenderme siempre.

A mi novia Dirisbel, por estar a mi lado compartiendo buenos y malos momentos.

A mis hermanos Jorge Luis Triana y Larissa Castillo, que son mi razón de ser.

A mi Tío Tatica, por guiarme y ser mí segundo padre.

A mi familia por siempre confiar en mí y estar a mi lado.

A todos mis amigos por su amistad incondicional, por apoyarme en todo momento y en especial a Enrique Expósito Y Sandor Escobar.

A todos los profesores que han contribuido a mi formación profesional.

A todos los que confiaron en mí.

DEDICATORIA

A mi madre, lo más grande del este mundo, sin ti nada sería posible.

A mi hermana que es el brillo de mis ojos.

A mi padre que estuviese orgulloso de mí y me sirvió de guía.

A mis abuelas por quererme tanto.

A toda mi familia que me ha dado su apoyo.

A todos mis amigos, esos hermanos que te van dando la alegría infinita.

RESUMEN

Se hace necesaria la utilización de los Sistemas de Información Geográfica (SIG) para el trabajo sobre los terrenos de nuestro planeta. El proyecto SIG-Desktop que está presente en la Universidad de las Ciencias Informáticas (UCI), tiene inmerso el desarrollo de nuevas funcionalidades para la plataforma de escritorio GeoQ, actualmente presenta la desventaja de tener que repetir acciones que pudo haber realizado el usuario anteriormente. Por tanto, el objetivo del presente trabajo de diploma se basa en desarrollar la automatización de acciones de usuario donde se garantice la rapidez y usabilidad para GeoQ. El software se desarrolló en Qt para el perfeccionamiento de interfaces de usuario con un conjunto de patrones para el mejoramiento y la calidad de la herramienta. Una vez incorporado al sistema GeoQ le brindará al usuario la facilidad y la comodidad para el trabajo con cartografía.

PALABRAS CLAVE

Automatización, Cartografía, GeoQ, SIG

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

El origen de la informática es el perfeccionamiento de los cálculos matemáticos realizado a través de la historia de la humanidad. Con el tiempo se han desarrollado las calculadoras mecánicas, el sistema binario, las máquinas lógicas y más tarde los lenguajes de programación de computadoras. Las computadoras formadas por miles de circuitos electrónicos integrados entre sí, no piensan como un ser humano, pero tienen la capacidad de realizar, analizar e interpretar gran cantidad de información de manera muy rápida. Con el desarrollo de las computadoras se ha dado un impulso a las Tecnologías de la Informática y las Comunicaciones (TIC), que a través de dispositivos o accesorios basados en la computación y las telecomunicaciones se obtiene y se guarda todo tipo de información.

El hombre ha creado los Sistemas de Información Geográfica (SIG) para dar respuesta a la necesidad que existe de ubicación y control sobre los terrenos del planeta, además de almacenar, manipular y visualizar datos que tengan un componente geográfico. En Cuba existen instituciones y organizaciones que se destacan por el uso de los SIG, entre ellas se encuentran: GEOCUBA, el Ministerio de Ciencia, Tecnología y Medio Ambiente (CITMA) y las Fuerzas Armadas Revolucionarias (FAR), cada una de ellas orientada a diferentes esferas entre las que se pueden citar: minería, meteorología, transporte y salud.

En la evolución tecnológica alcanzada en los últimos años en diferentes ámbitos del país, el desarrollo de los SIG, así como su vinculación y adaptación con el software libre, ha experimentado un progreso notable; para ello se han creado instituciones fortalecedoras del campo de la informática y el estudio y aplicación de los SIG.

La Universidad de Ciencias Informáticas (UCI) es una de estas instituciones, pues presenta una infraestructura de producción, sustentada por los centros de desarrollo distribuidos por todas sus facultades. En la Facultad 6 está presente el Centro de Desarrollo de Geoinformática y Señales Digitales (GEySED), que forma parte de la infraestructura antes mencionada. En este centro se lleva a cabo el desarrollo de una plataforma para la representación y edición de cartografías e información geoespacial. Vigente en él, uno de los proyectos llamado SIG DESKTOP, cuyo objetivo principal consiste en llevar a cabo la personalización de un sistema genérico capaz de representar información geográfica en una aplicación de escritorio nombrada GeoQ. Este sistema facilita la edición y análisis para el trabajo con objetos espaciales, la cual es uno de los factores fundamentales para el usuario a la hora de realizar un proyecto. Se espera que estos procesos sean los más usables porque depende la calidad de los resultados con un menor desgaste. Sin embargo, presenta limitaciones para el

usuario, entre las que se encuentra el tener que repetir funciones una y otra vez en aras de realizar un mismo proceso. Lo que ralentiza y dificulta el mismo; trayendo como consecuencia obstinación, posibilidad de errores humanos y tiempo de respuestas mayores.

Partiendo de la situación anterior se tiene como **problema a resolver** que *con GeoQ el usuario no puede recordar/reproducir el tipo y orden de acciones que realiza, provocándose inconformidad en el uso monótono del sistema*.

Centrando la atención en *la Automatización de acciones de usuario para el procesamiento de datos* teniendo así el **Objeto de Estudio**, dentro del cual se enmarca la *Automatización de acciones de usuario para procesamiento de datos cartográficos en el sistema GeoQ,* como **Campo de Acción** de la investigación.

Como **Objetivo General** de este trabajo es necesario *Desarrollar la automatización de acciones de usuario*.

Con el fin de dar cumplimiento al objetivo ya mencionado se plantea como **Idea a defender**: Si se implementa la automatización de acciones de usuario se podrá reducir la inconformidad y el tiempo de respuesta del mismo en la utilización del sistema GeoQ.

Para dar solución al objetivo general del trabajo se han desglosado las siguientes **tareas de investigación**:

1. Elaborar la fundamentación teórica para la repetición de acciones en los SIG.
2. Presentan las herramientas para el desarrollo de la automatización de acciones de usuario para procesamiento de datos cartográficos en el sistema GeoQ.
3. Realizar una propuesta de solución para la automatización de acciones de usuario.
4. Construir la automatización de acciones de usuario en el sistema GeoQ.

Para la realización del trabajo se consideró algunos **Métodos Investigativos** ya conocidos. A continuación se mencionan algunos así como la forma en que se ponen de manifiesto en la investigación.

Métodos Teóricos:

- Análisis histórico - lógico: Se utilizó para el análisis histórico de los procesos de automatización de acciones y de las características de los principales SIG existentes en el mundo. Permite además realizar el estudio de determinados elementos necesarios para llevar a cabo la

investigación, así como su evolución y desarrollo, obteniendo de dicho estudio las herramientas para el desarrollo del sistema.

- Analítico - sintético: Se empleó para el análisis de las tendencias y tecnologías actuales de los SIG en el mundo. Se utilizó para encontrar lo desconocido con el fin de explicar el sentido de lo encontrado partiendo de sus bases objetivas y subjetivas

Métodos Empíricos:

- Entrevistas: Es un método especialmente adecuado cuando no se tiene una teoría exacta sobre el tema, mientras que por el contrario se está a la expectativa para conocer los nuevos puntos de vista que no se habían previsto. Se realizan a especialistas de la Universidad con el objetivo de recopilar y obtener información referente a las herramientas óptimas para la construcción del sistema.

Estructura del documento

El trabajo estará estructurado en 4 capítulos expuestos a continuación:

Capítulo 1: Fundamentación teórica. Se plantean los elementos teóricos que sustentan el problema a resolver y se realiza un análisis de repertorio de soluciones similares desarrolladas en otros sistemas.

Capítulo 2: Tendencias actuales y tecnologías a utilizar. Se explican las principales técnicas, lenguajes de programación y herramientas que se manejarán para la construcción de la solución propuesta; haciendo énfasis en los elementos utilizados para identificarlos en el trabajo de las mismas.

Capítulo 3: Presentación de la solución propuesta. Se expone la solución a elaborar así como el modelo de dominio, la identificación de los requisitos funcionales, el diagrama de clase del dominio y el de caso de uso del sistema. Se realiza también la descripción textual correspondiente a cada caso de uso.

Capítulo 4: Construcción de la propuesta de solución. Se exponen las especificidades de la propuesta de solución para el problema planteado, además los diagramas y modelos de análisis y diseño, y diagramas de colaboración y despliegue; así como los resultados de las pruebas realizadas a la aplicación.

CAPITULO 1: FUNDAMENTACION TEORICA

1.1 Generalidades

En el presente capítulo se abordan las principales bases conceptuales para el mejor entendimiento y compresión de la investigación. Se realiza un resumen de los elementos esenciales del estado del arte de las tecnologías SIG y un análisis de los procesos participantes. Se tiene también presente las limitaciones de las soluciones existentes actualmente y la forma que dan respuesta al problema de la automatización de acciones.

1.2 Definiciones abordadas en la investigación

A continuación se exponen los principales conceptos y definiciones relacionados con el objeto de la investigación.

1.2.1 Tecnologías de la informática y las comunicaciones

Las tecnologías de la informática y las comunicaciones (TIC) se encargan del estudio, desarrollo, implementación, almacenamiento y distribución de la información mediante la utilización de hardware y software como medio de sistema informático. **(TICS, 2011)**

Es un conjunto de tecnologías que permite la adquisición, producción, almacenamiento, tratamiento, comunicación, registro y presentación de informaciones, en forma de voz, imágenes y datos contenidos en señales de naturaleza acústica, óptica o electromagnética. Las TIC incluyen la electrónica como tecnología base que soporta el desarrollo de las telecomunicaciones, la informática y el audiovisual. **(Grupo de Tecnología de la Informática y las Comunicaciones, 2011)**

Las tecnologías de la informática y las comunicaciones (TIC) plantean nuevos y eficientes paradigmas que revolucionan la vida actual. Constituyen en estos momentos, la mayor fuente de información existente facilitando el progreso de la sociedad humana mediante el uso, desarrollo, reserva y distribución de la misma.

1.2.2 Automatización

La automatización es un sistema donde se trasfieren tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos. **(Tecnológico, 2012)**

Automatizar es realizar procesos o trabajos utilizando poco o nada la mano del hombre. Existen cada vez más procesos automáticos, de un tipo u otro, incluso algunos de ellos no lo parecen a simple vista. **(Scribd, 2011)**

La automatización brinda una ayuda al usar la capacidad de las máquinas para llevar a cabo determinadas tareas anteriormente efectuadas por seres humanos. Controla la secuencia de las operaciones sin intervenir las personas dando un mayor control y fácil acceso. Los sistemas automatizados se utilizan para efectuar diversas tareas con más rapidez o mejor lo que ofrece un mayor rendimiento a la hora de realizar un trabajo.

1.2.3 Sistema de Información Geográfico

Un Sistema de Información Geográfico (SIG) consiste en información de naturaleza diversa sobre un determinado territorio, cuya relación se realiza a través de un sistema de referencia geográfico y se gestiona a través de uno o varios programas informáticos específicos. El conjunto es soportado por un sistema de computadoras y por un personal especializado. **(Grupo de Geomorfología, Hidrogeología y Medio Ambiente, 2011)**

Un SIG, es un sistema de gestión de bases de datos que permite mapear, integrar y analizar información geográfica para resolver problemas en investigación, planificación, ordenamiento y gestión geográfica. El SIG permite integrar distintos tipos de información según sean fotos aéreas, mapas, imágenes de satélite o según sea el nivel de definición en el que se trabaje, el nivel local, regional o nacional sobre la base de una extensión geográfica común y a su vez posibilitando la comunicación de resultados mediante mapas. **(Scribd, 2011)**

Los SIG poseen la capacidad de construir modelos o representaciones del mundo real constituyendo una valiosa herramienta para el análisis de diferentes tipos de fenómenos. La tecnología de los SIG puede ser utilizada para investigaciones científicas, la gestión de los recursos, la arqueología, la evaluación del impacto ambiental, la planificación y el manejo urbano, la cartografía, la sociología y la logística, por solo nombrar algunos.

Por ejemplo, un SIG podría permitir la representación de terrenos en caso de un desastre natural. Un SIG podría ser usado para identificar determinadas zonas geográficas que necesitan protección medioambiental, o podrían ser utilizados por una empresa para ubicar un nuevo negocio y aprovechar las ventajas de una zona de mercado con escasa competencia.

1.2.4 Cartografía

El desarrollo de las tecnologías y el conocimiento de construcción de mapas dieron lugar a la ciencia cartográfica. El hombre ya no quería pintar mapas, sino realizar obras científicas con extremada precisión. En la actualidad la cartografía es la ciencia que se ocupa de la preparación y construcción de los mapas y cartas náuticas, reproduciendo en una superficie plana la superficie terrestre.

Otras formas de ver la cartografía es la técnica de representar en forma convencional parte o toda la superficie terrestre sobre un plano utilizando; para este fin un sistema de proyección y una relación de proporcionalidad (escala) entre terreno y mapa. **(Sitio Web del Académico, 2011)**

La cartografía no es más que la ciencia que estudia los mapas geográficos y su trazado y dimensiones lineales. Es importante en la ubicación y la orientación del ser humano para el desarrollo de su conocimiento del espacio geográfico, sus formas de organización y el uso del mismo. Además de representar los contornos de objetos, las superficies y los ángulos; se ocupa también de representar la información que aparece sobre el mapa, según se considere qué es relevante y qué no. Esto, normalmente, depende de lo que se quiera representar en el mapa y de la escala. Actualmente el margen de error en los mapas es virtualmente nulo, ya que las técnicas actuales han hecho de la cartografía una ciencia exacta.

1.3 Análisis de las soluciones existentes

El hecho de tener que repetir operaciones ya realizadas por el usuario durante el trabajo con un SIG como pueden ser la adición, eliminación, colores de bordes y relleno; constituye una dificultad y una pérdida de tiempo en los mismos. Para facilitar el desarrollo de una herramienta que permita automatizar estas acciones se analizarán una serie de sistemas con características similares.

1.3.1 gvSIG Desktop

El gvSIG es un SIG de escritorio de código abierto con una licencia GNU/GPL (*General Public License*) capaz de acceder tanto a formatos vectoriales, como ráster, cuenta con un amplio número de herramientas para trabajar. Este sistema cuenta con SEXTANTE, una biblioteca libre que se distribuye bajo una licencia MIT (*Massachusetts Institute of Technology*), integrado en gvSIG, e incluso en otros SIG de escritorio. SEXTANTE tiene un gestor de geoproceso, una línea de comandos, un administrador de procesamiento por lotes, un generador de modelos y un historial de los comandos ejecutados por el usuario para que el proceso se pueda repetir fácilmente. Ver figura 1. **(gvSIG, 2011)**

SEXTANTE se compone de cuatro interfaces de usuario, en una de las cuales se ejecutan un elemento de la biblioteca y se agrega al historial. Dentro del Administrador de Historial de comandos el historial puede ser recorrido y las acciones individuales o bloques enteros de acciones pueden repetirse.

Teniendo en cuenta que SEXTANTE con la extensión en gvSIG brinda gran cantidad de funcionalidades, aparte de las herramientas que este presenta es capaz de repetir acciones y realizarlo con eficiencia; por otro lado es incapaz de recordar las acciones comunes realizadas por el mismo sistema gvSIG, por lo que se anula la utilización de esta solución para resolver el problema de esta investigación.

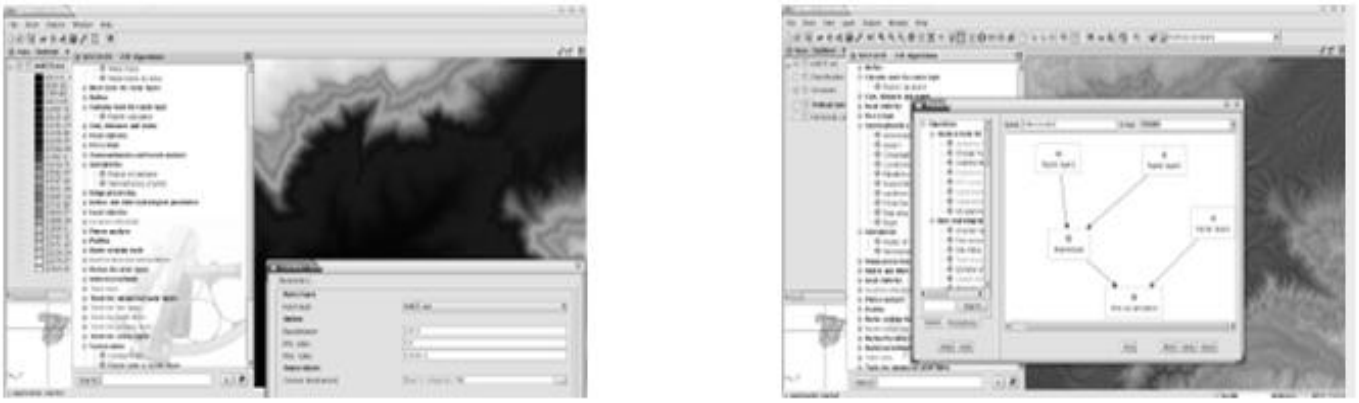

Figura 1 Librería SEXTANTE.

1.3.2 ArcGis Desktop

Este SIG de escritorio, de gran reconocimiento a nivel mundial, incluye un conjunto de aplicaciones integradas (ArcReader, ArcMap, ArcCatalog, ArcToolbox, ArcScene y ArcGlobe), además de diversas extensiones. ArcGIS Desktop se distribuye comercialmente bajo tres niveles de licencias que son, en orden creciente de funcionalidades y coste: ArcView, ArcEditor y ArcInfo. Sin embargo, no presenta una automatización de acciones de usuario para el trabajo con los datos. El documento Practical GIS Analysis, plantea un conjunto de ideas para los SIG y en ellas se encuentra la repetición de operaciones. **(Verbyla, 2003)**

1.3.3 Adobe Photoshop

Adobe Photoshop es el nombre o marca comercial oficial que recibe uno de los programas más populares de la casa Adobe Systems. Su capacidad de retoque y modificación de fotografías le ha dado el rubro de ser el programa de edición de imágenes más famoso del mundo.

La filosofía de trabajo del Photoshop se basa en el uso de capas que facilitan al usuario realizar cambios y modificar cualquier detalle en ellas a partir de información ubicada en otra capa cualquiera. Permite además, guardar acciones y con estas acciones guardadas se puede realizar cambios en

otras capas, con el objetivo de facilitar las operaciones con el usuario. Teniendo presente que Photoshop no es un SIG se usa como referencia por la similitud como herramienta de edición y manipulación gráfica. Por estas razones esta herramienta es bastante parecida a lo que se necesita para darle solución al problema.

1.4 Conclusiones parciales

Después de los análisis realizados así como las búsquedas bibliográficas e informativas se encontraron respuestas parciales a problemas similares brindadas por otras aplicaciones. Estas respuestas ofrecen una visión de los actuales SIG, de su incapacidad de repetir acciones por si solos. Proporciona una gran información para la creación sobre detalles visuales y datos que se gestionan como la biblioteca SEXTANTE y Photoshop, brindando una perspectiva a seguir. Concluyendo que hoy día no existe, ninguna solución capaz de automatizar la repetición de acciones en los SIG, por lo que el desarrollo del objetivo de esta investigación proporcionaría la primera de su tipo y GeoQ solucionaría la repetición de acciones y daría una mejor eficiencia en el trabajo con cartografías.

CAPITULO 2: TENDENCIAS ACTUALES Y TECNOLOGIAS A UTILIZAR

2.1 Introducción

En el presente capítulo se seleccionan las principales tecnologías a utilizar en la aplicación; los lenguajes de programación, las herramientas para la construcción, así como la metodología a seguir para el desarrollo de la investigación.

2.2 Lenguaje de Programación

Un lenguaje de programación es una técnica de comunicación estandarizada para expresar las instrucciones a una computadora. Se trata de un conjunto de reglas sintácticas y semánticas utilizadas para definir los programas de ordenador. (Cueva Lovelle, 1998)

Los lenguajes de programación permiten la creación de un software. Mejorando el desarrollo de las tareas para aplicaciones informáticas, estos lenguajes presentan formas que pueden ser leídas y escritas por las personas constituyendo, de esta forma, un modo práctico para dar instrucciones a un equipo.

Todos los lenguajes traen un conjunto de características para la mejora de su trabajo como son la eficiencia, el cúmulo de librerías y *frameworks* que disponen y si son multiplataforma. Partiendo que se utiliza C++ para el desarrollo de la plataforma GeoQ, se sigue con el mismo lenguaje para la implementación de la herramienta de automatización de acciones. Llegando a la conclusión que C++ es el lenguaje ideal y completamente compatible para el trabajo con la solución.

2.2.1 Lenguaje C++

El C++ es un lenguaje versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto en el desarrollo de aplicaciones. C++ mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original. **(Jalón, 1998)**

Es un lenguaje de programación orientado a objetos que puede ser usado de muchas maneras en el desarrollo de software. Es uno de los lenguajes más potentes ya que permite trabajar tanto a alto nivel como a bajo nivel. Se le han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, sobrecarga de operadores, referencias, operadores para el manejo de memoria persistente.

2.3 Entorno Integrado de Desarrollo

Un Entorno Integrado de Desarrollo (IDE) es un programa compuesto por un conjunto de herramientas para un programador, es decir, consiste en un editor de códigos, un compilador, un depurador y un constructor de interfaces gráficas. Los IDEs proveen un marco de trabajo amigable para los lenguajes de programación, tales como C++, Java, C#, Basic, Object Pascal, etc; y pueden soportar uno o varios de ellos. Los IDEs tienen como objetivo principal acortar la brecha entre el usuario y el lenguaje de programación.

Teniendo al alcance un IDE de desarrollo como Qt Creator, que brinda grandes facilidades para la creación de aplicaciones el cual presenta completamiento de código y una ayuda lo bastante abundante que sirve de guía. Se enfatiza que la plataforma GeoQ esta implementada con Qt Creator. Llegando a la conclusión que se utiliza este mismo IDE de desarrollo para la solución de la investigación.

2.3.1 Qt Creator como IDE de desarrollo

Qt Creator es un IDE moderno, fácil de manejar, abierto, gratuito y multiplataforma que permite desarrollar aplicaciones complejas para escritorio y plataformas móviles. Está completamente orientado a objetos, permite realizar programación visual y programación dirigida por eventos. Se basa en la librería Qt y cuenta con las siguientes características principales:

- Editor avanzado para C++.
- Diseñador de formularios (GUI) integrado.
- Herramientas para la administración y construcción de proyectos.
- Completado automático.
- Depurador visual.

2.4 Herramientas CASE

Las Herramientas CASE (Ingeniería de Software Asistida por Ordenador) en inglés (*Computer Aided Software Engineering*) supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información. Es un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software (investigación preliminar, análisis, diseño, implementación e instalación).

El Visual Paradigm, es una herramienta CASE desarrollada con tecnologías libres, factor fundamental que garantiza la soberanía e independencia tecnológica, multiplataforma, posibilita la creación de diagramas para UML. Brindaría un aumento a la calidad del software y mejora la productividad en el desarrollo y mantenimiento del mismo.

Para darle solución a la problemática se utilizará la versión de Visual Paradigm: 5.0

2.4.1 Visual Paradigm como herramienta CASE

Visual Paradigm permite representar todo tipo de diagramas UML de la versión 2.0 para las distintas fases por las que transita un software en desarrollo: análisis, diseño e implementación. Sus características principales son:

- Disponible en múltiples plataformas.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.

2.5 El lenguaje de Modelado (UML)

UML es un lenguaje estándar para crear planos de software, no es un lenguaje de programación; sin embargo, permite hacer una rápida transición del modelo al código. El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la implementación y configuración con los diagramas de despliegue.
El UML es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90's. UML es un lenguaje de modelado, no un método. El UML, fusiona los conceptos de la orientación a objetos aportados por Booch, OMT y OOSE. **(Booch, Rumbaugh, & Jacobson, 2000)**

Este estándar de modelado presenta un grupo de ventajas en las que se puede encontrar apoyo:

- Es adaptable a casi cualquier sistema.
- Puede ser utilizado en la mayoría de fases de un proyecto.
- Permite modelar estructuras complejas.

- Admite cubrir las vistas necesarias para desarrollar y luego desplegar los sistemas.

2.6 Metodologías de desarrollo de software

"Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software." **(Carrillo Pérez, Pérez González, & Rodríguez Martín, 2008)**

Metodología ligera

Las metodologías orientadas al intercambio directo con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Estas son llamadas Metodologías ligeras/ágiles. **(Carrillo Pérez, Pérez González, & Rodríguez Martín, 2008)**

Metodología pesada

Se centra en la definición detallada de los procesos y tareas a realizar y requiere una extensa documentación, ya que pretende prever todo de antemano. Este tipo de metodologías son más eficaces y necesarias cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que son necesarios emplear, donde una gran organización es requerida. **(Carrillo Pérez, Pérez González, & Rodríguez Martín, 2008)**

Las metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas metodologías son llamadas Metodologías pesadas.

Para la realización de la investigación se propone como metodología RUP (Rational Unified Process), teniendo en cuenta que el sistema GeoQ se basa en esta metodología, garantizando que con su aplicación se generan la documentación y artefactos imprescindibles para que los futuros clientes tengan como base una guía para el entendimiento.

2.6.1 Proceso Unificado Racional

Una de las metodologías pesadas más conocidas es la Metodología RUP que divide el desarrollo en 4 fases que definen su ciclo de vida: **(Tobarra Narro, 2003)**

- Inicio: El objetivo es determinar la visión del proyecto y definir lo que se desea realizar.
- Elaboración: En esta fase se determina la arquitectura óptima del proyecto.

- Construcción: Se obtiene la capacidad operacional inicial.
- Transmisión: Obtener el producto acabado y definido.

Entre las características que distinguen a RUP sobresalen en la investigación:

Dirigido por casos de uso: Los casos de uso reemplazan la antigua especificación funcional tradicional y constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema. **(Martínez, y otros, 2004)**

Centrado en la arquitectura: Establecer una arquitectura candidata al inicio es un paso muy importante, pues será ella la que guíe el desarrollo del sistema. Esto permitirá el desarrollo en paralelo, minimizar la repetición de trabajos e incrementar la probabilidad de reutilización de componentes. **(Martínez, y otros, 2004)**

Iterativo e incremental: El desarrollo iterativo ofrece la posibilidad de que los elementos sean integrados continuamente, facilita el rehuso y resulta un producto más robusto debido a que los errores son corregidos en cada iteración. **(Martínez, y otros, 2004)**

2.7 Conclusiones parciales

En este capítulo se analizaron las herramientas y tecnologías a utilizar en el desarrollo de la automatización de acciones de GeoQ. El objetivo fundamental de la selección de las herramientas es escoger las adecuadas a utilizar basándose en las características de las mismas y las mejorías que estas atribuyen al sistema GeoQ. Teniendo como conclusión el empleo del lenguaje C++ utilizando como IDE a Qt Creator. La propuesta de solución a desarrollar estará bajo la tutela de la metodología RUP y los diagramas serán modelados utilizando el instrumento Visual Paradigm de CASE.

El conocimiento de las ventajas que traería el uso de las herramientas y tecnologías antes expuestas para el desarrollo de la aplicación constituye una guía necesaria para un mejor trabajo, permitirá la obtención de un mayor rendimiento y un menor costo de producción del producto final. Luego de todo lo detallado anteriormente se puede afirmar que las bases para comenzar la construcción, en términos ingenieriles, de la futura aplicación están debidamente instauradas.

CAPITULO 3: PRESENTACION DE LA SOLUCION PROPUESTA

3.1 Introducción.

En el presente capítulo se aborda los temas relacionados con el análisis y el diseño de la solución propuesta. Se realiza el modelo de dominio para una mayor comprensión del contexto en que se desarrolla el sistema además del levantamiento de los requisitos, tanto funcionales como no funcionales, con que debe contar el mismo, el diagrama de casos de uso del sistema así como la descripción textual de los mismos y los diagramas de clases del diseño.

3.2 Modelo del Dominio.

En el estudio realizado de acuerdo con el problema a resolver no fue posible identificar los procesos del negocio, debido a que se hace difícil determinar los elementos más importantes que intervienen, por estas razones se propone realizar un modelo conceptual o modelo de dominio.

Un Modelo del Dominio (MD) es una representación de las clases conceptuales del mundo real, no de componentes software. Este modelo no se trata de un conjunto de diagramas que describen clases de software u objetos de software con responsabilidades, sino que puede considerarse como un diccionario visual de las abstracciones relevantes, vocabulario e información del dominio. El MD se centra en una parte del negocio, la relacionada con el ámbito del proyecto y representa una actividad clásica del análisis orientado a objetos. Aprovechando las bondades de los diagramas UML para representar conceptos, el MD se presenta en forma de diagrama de clases donde figuran los principales conceptos y roles del sistema en cuestión.

3.2.1 Definición de clases del modelo de dominio.

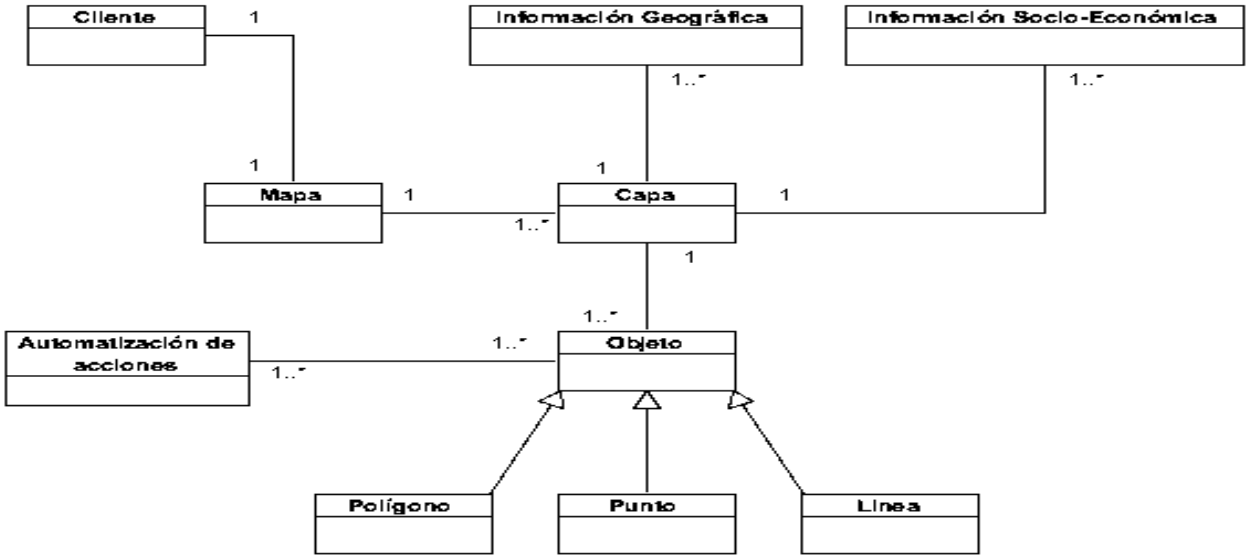

Figura 2 Diagrama del Modelo de Dominio.

Cliente: Persona que necesita utilizar el software para trabajar en la repetición de acciones.

Mapa: Representación gráfica y métrica de una porción de territorio.

Capas: Define la información a representar en el mapa.

Objeto: Elemento contenido en una capa para brindar información al cliente.

Línea: Objeto que contiene una capa.

Punto: Objeto que contiene una capa.

Polígono: Objeto que contiene una capa.

Información geográfica: Datos espaciales georreferenciados utilizados en operaciones administrativas.

Información socio-económica: Datos espaciales georreferenciados utilizados en operaciones social y económico.

Automatización de acciones: Proceso que desea realizar el cliente para repetir acciones partiendo de los objetos (línea, punto, polígono) que se encuentren en una capa.

3.3 Requisitos

De acuerdo con el Proceso Unificado de Rational, el término requisito puede definirse como una condición que el sistema debe cumplir o capacidad que debe tener. Para que un requisito se considere

satisfactorio debe cumplir las siguientes características: implementación independiente, consistente y no ambiguo, preciso, verificable, que pueda ser leído y modificable.

Cuanto más grande e intrincado sea el sistema en desarrollo, más tipos de requisitos aparecen cuando se inicia la recolección de estos. Mediante la identificación de los tipos de requisitos, los equipos de desarrollo de software pueden separar grandes cantidades de requisitos en grupos que faciliten su manejo, también se logra una comunicación más clara entre los miembros del equipo, y en general se mejora el manejo del proyecto en su totalidad.

Existen dos grandes categorías en las que pueden clasificarse los requisitos, estas son: requisitos funcionales y requisitos no funcionales.

3.3.1 Requisitos Funcionales

Los requisitos funcionales expresan una especificación detallada de las responsabilidades de la herramienta que se propone. Ellos permiten determinar, de una manera clara, lo que debe hacer el software.

RF1: Recopilar información de navegación.

RF2: Recopilar información de tematización.

RF3: Recopilar información de edición.

RF4: Ejecutar lista de acciones.

3.3.2 Requisitos No Funcionales

Los Requisitos No Funcionales (RNF) son propiedades o cualidades que el producto debe tener. Normalmente, están vinculados a requisitos funcionales, es decir, una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. **(Pressman, 2005)**

Usabilidad

- El sistema estará destinado para desarrolladores de proyectos con mapas. Los usuarios deben poseer un conocimiento mínimo de la herramienta.
- El sistema debe poseer una interfaz amigable, de fácil acceso y manipulación, que facilite guardar y cargar las diferentes funcionalidades presentes en ella.

Apariencia o interfaz externa

- El sistema debe tener una apariencia profesional con un diseño gráfico sencillo de tonalidades de colores claros que facilite la localización de las funciones.
- La interfaz debe ser de fácil comprensión en su funcionamiento permitiendo la utilización sin mucho adiestramiento.
- La herramienta debe contar con el nombre de la aplicación en la parte superior mediante una barra de título así como el nombre de cada una de las ventanas en dependencia de la funcionalidad que esté presente.

Portabilidad y operatividad.

- La aplicación debe ser compatible con los Sistemas Operativos Microsoft Windows XP, en plataformas Unix/Linux.

Software

Las computadoras que utilizarán el software deben tener instalado:

- Microsoft Windows XP Profesional, una distribución de Linux de la familia Debian (Debian, Ubuntu, Nova).

Hardware

- Las computadoras que utilizarán la aplicación deberán contar con un microprocesador con velocidad de procesamiento superior a 1 GHz y memoria RAM de 512 MB o superior.

3.4 Casos de Uso del Sistema

Luego de realizado un estudio de la descripción de los requisitos, que constituyen las funcionalidades esenciales del sistema, se pasa a la siguiente etapa de construcción de un software, la descripción del modelo de casos de uso del sistema. El modelo de casos de uso del sistema describe la funcionalidad propuesta del nuevo sistema. Un caso de uso (CU) representa una unidad discreta de interacción entre un usuario (humano o máquina) y el sistema. Un modelo de casos de uso está compuesto por actores, casos de uso y las relaciones existentes entre ellos.

3.5 Diagrama de Caso de uso del Sistema

A continuación se muestra el diagrama de caso de uso del sistema donde se representan los actores y su relación con el sistema:

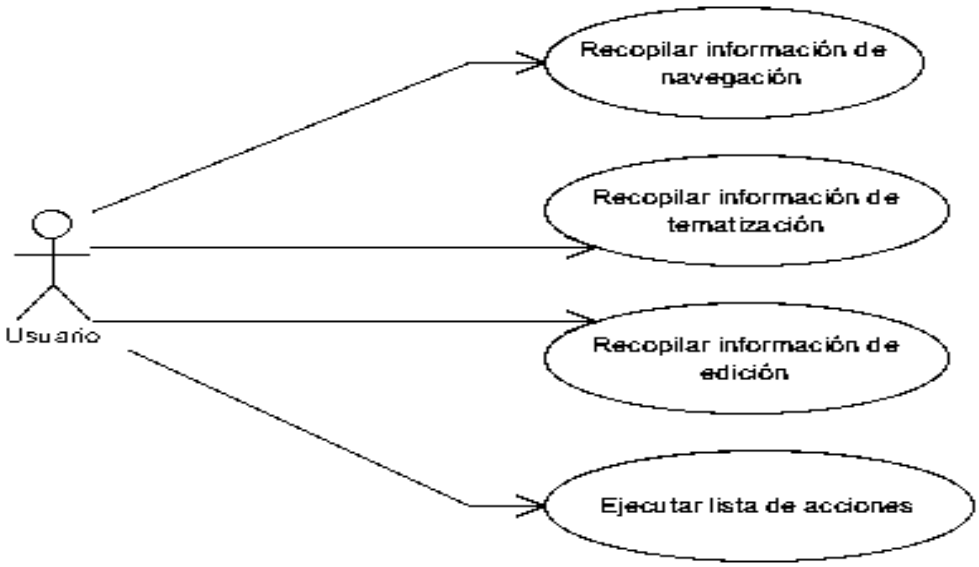

Figura 3 Diagrama de Caso de uso del Sistema.

3.6 Descripción textual de los casos de uso del Sistema

Con la representación gráfica del diagrama de casos de uso no es suficiente para lograr entender las funcionalidades asociadas a un CU, por lo que es necesario describir textualmente cada uno de estos, de manera que queden especificadas todas las acciones necesarias que realizan el actor y el sistema. Con la descripción que se presentan a continuación de un CU, se obtendrá una idea más clara de cómo se realiza el proceso de automatización y quiénes intervienen directamente en este. Ver Anexos para las otras descripciones.

Caso de Uso:	Recopilar información de edición
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario desea guardar acciones. El caso de uso termina cuando el usuario detiene el proceso de guardar acciones.
Referencias	RF3
Prioridad	Alta
Precondiciones	Debe estar cargada alguna cartografía y estar activada la opción Macro Recording.
Flujo de eventos	

Recopilar información de edición		
Actor		**Sistema**
	1- Presiona el botón iniciar de la ventana Macro Recording.	2- El sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El sistema brinda las opciones de Seleccionar, Mover objeto, Agregar objeto, Eliminar objeto, Copiar objeto, Cortar objeto, Pegar.
	3- El usuario comienza a realizar las acciones. Las acciones a realizar son: - Seleccionar - Mover objetos - Agregar objeto - Eliminar objeto - Copiar - Cortar - Pegar	4- El sistema responde en correspondencia a lo que el usuario seleccionó: -Si el usuario selecciona la opción de Seleccionar. Ver sección Seleccionar. -Si el usuario selecciona la opción de Mover objetos. Ver sección Mover objetos. -Si el usuario selecciona la opción de Agregar objeto. Ver sección Agregar objeto. -Si el usuario selecciona la opción de Eliminar objeto. Ver sección Eliminar objeto. -Si el usuario selecciona la opción de Copiar. Ver sección Copiar. -Si el usuario selecciona la opción de Cortar. Ver sección Cortar. -Si el usuario selecciona la opción de Pegar. Ver sección Pegar.
Sección : “Seleccionar”		
	Actor	**Sistema**
		5- El sistema cambia la forma del puntero.
	6- El usuario selecciona el área deseada.	7- El sistema incorpora la acción en el panel de control.

Prototipo de Interfaz

Sección : "Eliminar objeto"		
	Actor	**Sistema**
		5- El sistema muestra los objetos seleccionados para que el usuario pueda eliminarlos y muestra una ventana emergente para confirmar la eliminación.
	6- El usuario presiona el botón Aceptar	7- El sistema incorpora la acción en el panel de control.

Prototipo de Interfaz

Flujos alternos		
	Actor	**Sistema**

	6.1- El usuario presiona el botón Cancelar	6.2-El sistema no realiza cambios en el panel de control.
Prototipo de Interfaz		
Sección : "Mover objeto"		
	Actor	**Sistema**
	5- El usuario realiza el movimiento de los objetos seleccionado al lugar deseado.	6- El sistema incorpora la acción en el panel de control.
Prototipo de Interfaz		
Sección : "Copiar objeto "		
	Actor	**Sistema**
		5- El sistema almacena los objetos copiados.
Sección : "Cortar objeto "		
	Actor	**Sistema**
		5- El sistema incorpora la acción en el panel de control.
Prototipo de Interfaz		

<table>
<tr><td colspan="3">
</td></tr>
<tr><td colspan="3">Sección : “Pegar objeto ”</td></tr>
<tr><td></td><td>Actor</td><td>Sistema</td></tr>
<tr><td></td><td></td><td>5- El sistema muestra los objetos pegados e incorpora la acción en el panel de control.</td></tr>
<tr><td colspan="3">Sección : “Agregar objeto ”</td></tr>
<tr><td></td><td>Actor</td><td>Sistema</td></tr>
<tr><td></td><td>5- El usuario agrega los objetos en el lugar deseado.</td><td>6- El sistema incorpora la acción en el panel de control.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz
</td></tr>
<tr><td>Postcondiciones</td><td colspan="2">Se almacenó la información de las acciones realizadas por el usuario en el panel de control.</td></tr>
</table>

Tabla 1 Descripción del CU “Recopilar información de edición”.

3.7 Conclusiones Parciales

En este capítulo se determinaron las características básicas con las que debe contar un sistema para poder utilizar el producto. Se definieron cuatro requisitos funcionales, con los cuales se confeccionó el diagrama de caso de uso del sistema. Además, se consumó la descripción textual de un CU, para lograr un mejor entendimiento de las funcionalidades de la herramienta a desarrollar. Concluyendo que el software a desarrollar es de fácil utilización y brinda una importante documentación de los artefactos, finalmente, es soportado por la mayoría de las computadoras con que cuenta la sociedad cubana de la actualidad.

CAPITULO 4: CONSTRUCCION DE LA SOLUCION PROPUESTA

4.1 Introducción

En este capítulo se detalla la construcción de la solución propuesta, a través del Análisis se refinan los requerimientos, además de mostrar el interior y la conformación del sistema. Los flujos de Diseño e Implementación, que tiene como objetivo principal transformar los requerimientos en un diseño consecuente con la implementación del sistema, diseñando hacia un enfoque al rendimiento, así como establecer una línea base de la arquitectura que soporte a los requisitos del sistema. Mediante los diagramas se tiene una guía vital para el desarrollo correcto de la aplicación. Se define además, el modelo de despliegue originado por la selección de los artefactos más importantes para el sistema donde se precisan los componentes que conforman la estructura física de la aplicación.

4.2 Análisis y Diseño

En este flujo se refinan los requisitos y se razona sobre los aspectos internos del sistema. En el Modelo de Sistema se definen los requisitos capturados por Casos de Uso, agrupando estos, las funcionalidades del software a implementar. Se introduce un mayor formalismo, pues se realizan descripciones utilizando el lenguaje de los desarrolladores: proporcionando así una visión general del sistema. Este es uno de los flujos de trabajo de software indicado por RUP, necesario para investigar y definir el comportamiento de la solución informática a desarrollar como una caja negra (el comportamiento del sistema). Describe lo que hace el software, sin explicar la manera que lo hace. Una parte de la descripción, son los diagramas de colaboración del sistema, los cuales muestran cómo interactúan cada una de las clases identificadas en este flujo a partir de los requerimientos detectados. **(Larman, 1999)**

4.3 Modelo de Análisis

El lenguaje utilizado en el análisis se basa en un modelo de objetos, llamado *modelo de análisis*, el cual cuenta con clases que ayudan a definir las partes internas del sistema. Estas clases interactúan entre sí, para realizar cada una de las funcionalidades necesarias. En resumen, el modelo de análisis ayuda a estructurar y refinar los requisitos, además de brindar una vista interna del sistema.

En el modelo de análisis se muestran los conceptos básicos del sistema, la abstracción más general que se puede hacer del mismo, sin tener en cuenta la plataforma o el lenguaje a desarrollar. Fue realizado a través de un diagrama de clases, en el que se representan las clases y las asociaciones preliminares entre ellas.

Una clase del análisis participa en relaciones, aunque estas son más conceptuales que sus contrapartidas de diseño e implementación. Las clases del análisis siempre encajan en uno de tres estereotipos básicos: de interfaz, de control o de entidad. Cada estereotipo indica una semántica específica, lo cual constituye un método potente y consistente de identificar y describir las clases del análisis y contribuye a la creación de un modelo de objetos y una arquitectura robusta.

A continuación, la figura 4 muestra el diagrama de clases del análisis realizado al CU “Recopilar información de edición”.

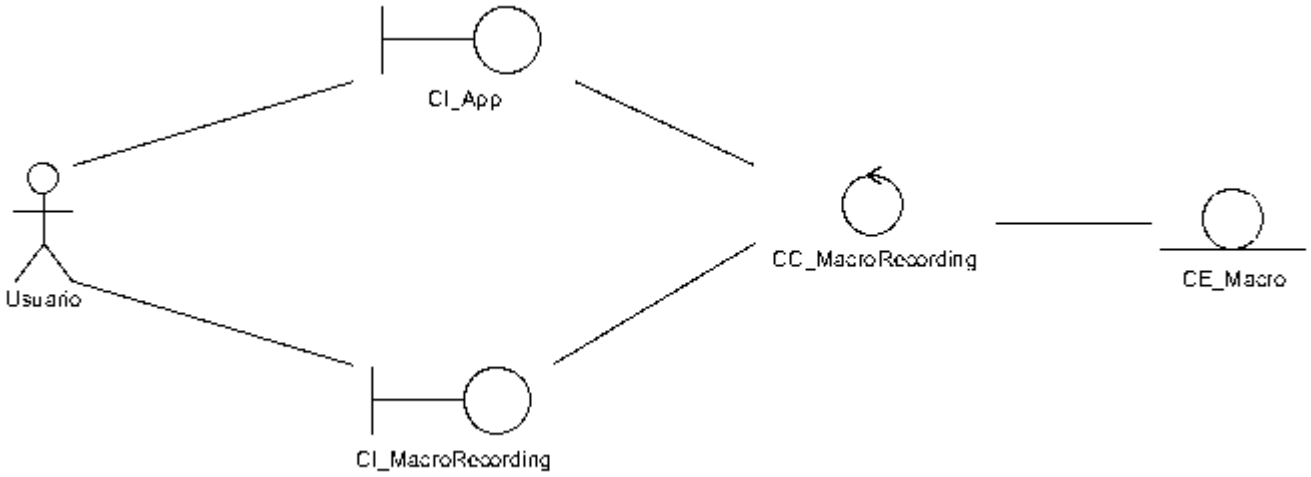

Figura 4 Diagrama de Clases del Análisis del CU “Recopilar información de edición”.

4.4 Patrones arquitectónicos

Un estilo arquitectónico denominado también patrón arquitectónico se define como una familia de sistemas de software en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores y un conjunto de restricciones de cómo pueden ser combinadas. Para muchos estilos puede existir uno o más modelos semánticos que especifiquen cómo determinar las propiedades generales del sistema partiendo de las propiedades de sus partes. **(Camacho, 2004)**

Los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de software. Proveen un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos. Proponen que son plantillas para arquitecturas de software concretas, que especifican las propiedades estructurales de una aplicación con amplitud de todo el sistema y tienen un impacto en la arquitectura

de subsistemas. La selección de un patrón arquitectónico es, por lo tanto, una decisión fundamental de diseño en el desarrollo de un sistema de software. **(Buschmann, 1996)**

En resumen se puede definir estilo arquitectónico, como un conjunto de reglas de diseño que identifican las clases de componentes y conectores que se pueden utilizar para componer un sistema o subsistema, junto con las restricciones locales o globales de la forma en que la composición se lleva a cabo.

Realizado el estudio de las características principales y el análisis de las definiciones de los patrones arquitectónicos, se determinó la aplicación de la arquitectura basada en componentes, y la implementación de patrones orientados a objetos, teniendo presente la utilización de estas mismas arquitecturas por el sistema GeoQ. A continuación se argumentan teóricamente los criterios asumidos para las selecciones anteriores y los principios básicos presentes en los mismos.

Arquitectura basada en componentes

La arquitectura basada en componentes describe una aproximación del diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas, dicha descomposición permite a su vez convertir los componentes de software ya existentes en piezas más grandes de software, esto da lugar al principio de reutilización del software y promueve que los componentes sean implementados de tal forma que permita luego su utilización funcional en diferentes sistemas. En la arquitectura basada en componentes también se requiere robustez debido a que los componentes han de operar en entornos heterogéneos.

El principal elemento de software dentro de una arquitectura basada en componentes es precisamente el componente de software: una unidad de composición que solo depende del contexto contractual de forma específica y explícita. **(Pressman, 2005)**

Por tanto se puede definir como principios fundamentales de un componente de software los siguientes:

Reusabilidad: Los componentes son normalmente diseñados para utilizarse en diferentes escenarios por distintas aplicaciones, no obstante, determinados componentes pueden ser diseñados para tareas específicas.

Extensibilidad: Los componentes pueden ser extendidos a partir de componentes ya existentes para así brindar un nuevo comportamiento.

Independencia: Los componentes son diseñados para tener una menor dependencia de otros componentes y estos pueden ser aplicados en un ambiente adecuado sin afectar otros componentes o sistemas.

Sin contexto específico: Los componentes son diseñados para operar en diferentes ambientes y contextos. Información específica como el estado de los datos deben ser pasadas al componente en vez de incluirlos o permitir al componente acceder a ellos.

Encapsulamiento: Los componentes exponen interfaces que permiten al programa usar su funcionalidad y no revelar los detalles de los procesos internos, detalles del proceso o estado.

Arquitectura orientada a objetos

El diseño de software orientado a objetos requiere la definición de una arquitectura de software multicapa, la especificación de subsistemas que realizan funciones necesarias y proveen soporte de infraestructura, una descripción de objetos (clases), que son los bloques de construcción del sistema, y una descripción de los mecanismos de comunicación, que permiten que los datos fluyan entre capas, subsistemas y objetos. El Diseño Orientado a Objetos (DOO), cumple todos estos requisitos. **(Pressman, 2005)**

Se puede determinar entre los principios claves de la arquitectura orientada a objetos: el encapsulamiento, herencia y polimorfismo debido a que el estilo se basa en principios orientados a objetos.

Encapsulamiento**:** Los objetos presentan funcionalidades solo mediante eventos, métodos y propiedades, además ocultan detalles internos como el estado y las variables de otros objetos. Esto propicia que sea más fácil reemplazar o actualizar los objetos, siempre y cuando sus interfaces sean compatibles, todo esto sin afectar otros objetos.

Herencia: Los objetos pueden heredar de otros objetos y emplear funcionalidades del objeto base o sustituirlas. La herencia a su vez permite que la actualización y el mantenimiento sean más sencillos, ya que los cambios en el objeto base se propagan automáticamente a los objetos heredados.

Polimorfismo**:** Permite sustituir el comportamiento de un tipo de objeto base que apoya las operaciones en la aplicación mediante la implementación de nuevos tipos que son intercambiables con el objeto existente.

4.5 Principios de Diseño

Los principios de diseño son una secuencia de pasos, no constituyen una receta pues intervienen: la creatividad, experiencia y un compromiso con la calidad donde existen factores internos (son buscados por el ingeniero de software) y externos (propiedades que pueden ser observadas por los usuarios).
En el libro “Ingeniería de Software Un enfoque práctico” de Pressman, Alan Davis define el principio de diseño como el proceso: **(Pressman, 2005)**

- Donde deben tomarse enfoques alternativos.
- Que deberá rastrearse hasta el análisis.
- Que se debe reutilizar.
- Que debe tratar de imitar el dominio del problema.
- Busca uniformidad e integración.
- Que deberá estructurase para admitir cambios.
- Para prevenir la adaptación a circunstancias inusuales.
- Que no debe codificar.
- Que debe ser evaluado en función de calidad mientras está creciendo.
- Que minimice errores conceptuales.

Como principios de diseño del sistema se tienen:

- Una iconografía descriptiva, es decir, que cada ícono se ajuste a la funcionalidad que represente.
- Mostrar al usuario un mensaje de confirmación, de manera que pueda asegurarse que la operación realizada es correcta.
- Brindar una interfaz sencilla, de manera tal que cualquier persona con un mínimo dominio de la computación pueda aprender a trabajar con la aplicación.
- Los nombres de los atributos comenzarán con letra minúscula al igual que los métodos.

4.6 Patrones de Diseño

"Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles". **(Booch, Rumbaugh, & Jacobson, 2000)**

Los patrones de diseño son el inicio en la búsqueda de soluciones en el desarrollo de software y otros ámbitos referente al diseño de interacción o interfaces. Para que la solución sea factible el patrón debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores y ser reusables, lo que significa que se puede aplicar a diferentes problemas de diseño en diferentes circunstancias.

Patrón Singleton

- Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

La clase QgsApp presente el patrón singleton con su única instancia dando un único acceso a objetos que esta presenta.

Patrón Observador

- Este patrón define una dependencia entre un objeto y un conjunto de ellos, de modo que los cambios en el primero se vean reflejados en los otros, es decir, cuando un objeto cambie de estado se notifica y actualizan automáticamente todos los objetos que dependen de él.

- Este patrón permite añadir nuevos observadores sin modificar al sujeto o a otros observadores. Que el sujeto no informe a sus observadores de que cambio ha sufrido permite mantener el acoplamiento en un nivel bajo, puesto que el observador solo pide los datos del estado del sujeto que le interesan.

El empleo del patrón observador se ve reflejado en un conjunto de clases como son QgsMapToolSelect, QgsMapToolAddFeature, QgsMapToolMoveFeature. Estas clases encargada de emitir señales modificando estados de otras clases como son QgsApp y MacroDialog.

Patrón Creador

- La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos.
- En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella.
- El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento.

Se toma el patrón creador, mostrando como ejemplos la clase MacroDialog, creando instancias de la clase QgsVectorLayer porque contiene objetos de ella que sirven de ayuda a la solución.

Patrón Bajo Acoplamiento

- Acoplamiento bajo significa que una clase no depende de muchas clases.

La clase MacroDialog presenta un bajo acoplamiento, no depende innecesariamente de otras clases. En la estructura y composición la clase es lo más independientes posible, facilitando su reutilización porque no requieren la presencia de muchas otras clases; y evita que un cambio en ella no afecte a otras.

Beneficios

- No se afectan las clases por cambios realizados a otros componentes.
- Fácil de entender por separado.
- Mejora la reutilización de las clases.

Patrón Comando

- Encapsular una petición o llamado como un objeto.
- Parametriza las colas o registros de llamados.
- Permitir operaciones de deshacer.

La clave de este patrón de diseño es una clase de comando abstracto, que declara una interfaz en la clase MacroRecording para ejecutar operaciones. Concretamente la subclase comando específica una pareja receptor-acción para almacenar el receptor como instancia de una variable.

Beneficios

- Mayor flexibilidad en el diseño de las interfaces gráficas. Una aplicación puede proporcionar una interfaz para un botón y una interfaz para un menú como una característica para hacer al menú y al botón compartir una instancia concreta de la subclase Comando. **(univalle, 2011)**
- Poder reemplazar los comandos dinámicamente, los cuales serían muy útiles para implementar menús contextuales sensitivos. Todo esto es posible porque el objeto que emite el llamado solo necesita conocer quien lo emite; este no necesita conocer cómo se llevara a cabo el llamado. **(univalle, 2011)**

4.7 Modelo de Diseño

El Modelo de Diseño es un modelo de objetos que describe la realización de los CU y pretende crear un plano del Modelo de Implementación y de código fuente, ya que es utilizado como una entrada esencial en las actividades de implementación y prueba. Se emplea además para concebir, en adición a la documentación, el diseño del sistema del software. De manera general, es un abarcador y compuesto artefacto que encapsula todas las clases de diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos.

Una clase de diseño representa una abstracción de una o más clases en la implementación de un sistema. Las clases definen objetos, los cuales a su vez implementan los CU; si las mismas son buenas o no dependen profundamente del entorno de implementación. De manera general, las clases deben mapearse en un fenómeno en particular: en el lenguaje de implementación y deben estar estructuradas de manera que representen los resultados en un buen código. Aun cuando las peculiaridades del lenguaje de implementación influyen en el modelo de diseño, en el presente trabajo de diploma se ha tratado de mantener una estructura de clases de fácil entendimiento y modificación.

A continuación, la figura 5 muestra el diagrama de clases del diseño realizado al CU “Recopilar información de edición”.

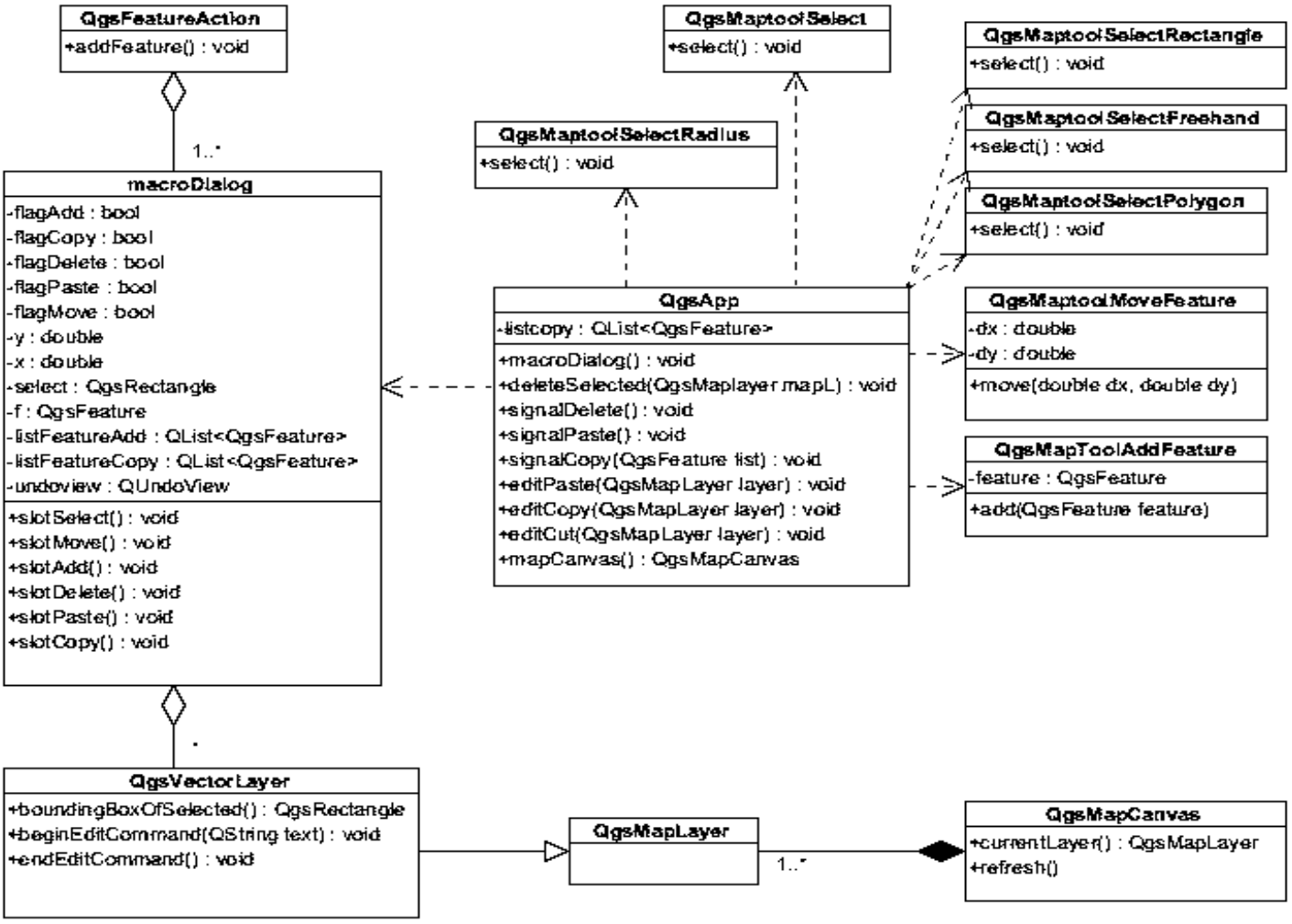

Figura 5 Diagrama de Clases del Diseño del CU "Recopilar información de edición".

4.8 Diagramas de interacción

Dentro de los artefactos que define el diseño para alcanzar sus objetivos se encuentran los diagramas de interacción. Estos especifican la secuencia de acciones que deben llevarse a cabo, entre los objetos que conforman el sistema, para realizar un CU. Los diagramas de interacción se pueden expresar en términos de secuencia y/o colaboración. El objetivo es transformar los requerimientos en un diseño del sistema que se desarrollará y adaptarlo para que sea efectivo en el ambiente de implementación.

4.9 Diagrama de secuencia

Un diagrama de secuencia muestra una interacción ordenada según la secuencia temporal de eventos, los objetos que participan en la interacción y los mensajes que estos intercambian ordenados según su secuencia en el tiempo. Para cada uno de los casos de uso del sistema se crea un diagrama de secuencia, interviniendo en estos los actores del CU y objetos representantes del sistema, mostrando los eventos que se envían unos a otros.

4.10 Diagrama de colaboración

Los diagramas de colaboración muestran una interacción organizada basándose en los objetos que toman parte en la interacción y los enlaces entre los mismos. La secuencia de los mensajes y los flujos de ejecución concurrentes se determinan explícitamente mediante números de secuencia. Se muestra a continuación el diagrama de colaboración realizado a uno de los CU. La figura 6 es la representación realizada al CU “Recopilar información de edición”.

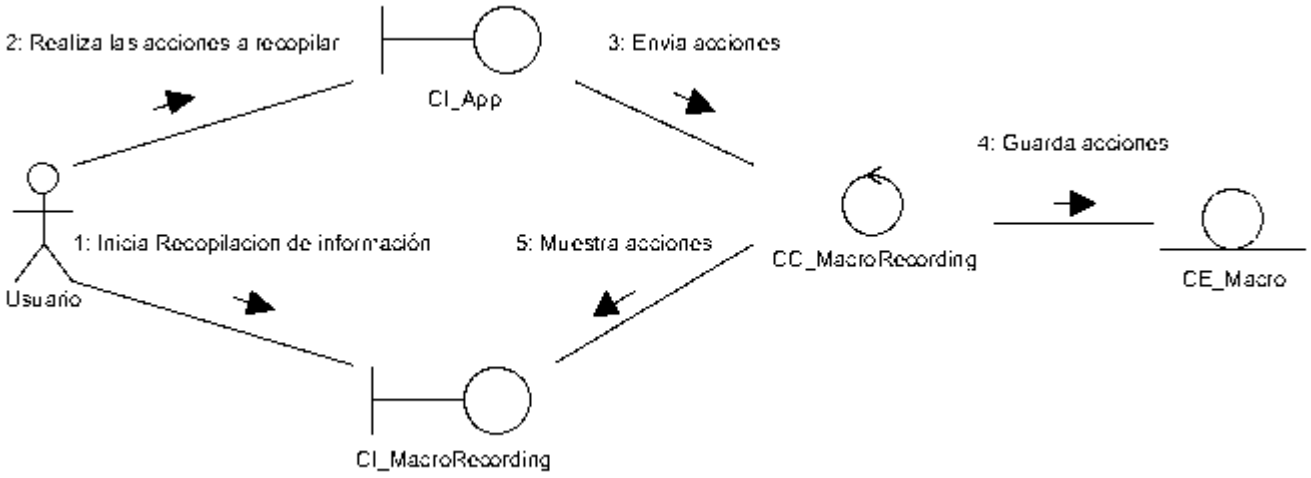

Figura 6 Diagrama de Colaboración del CU “Recopilar información de edición”.

4.11 Modelo de Componente.

El modelo de implementación es una visión general de lo que se debe implementar, que coincide con la plantilla del plan de integración, con los componentes y subsistemas a implementar, así como con los resultados que se deben obtener y las pruebas que se deben realizar sobre ellos.

4.11.1 Diagrama de componentes.

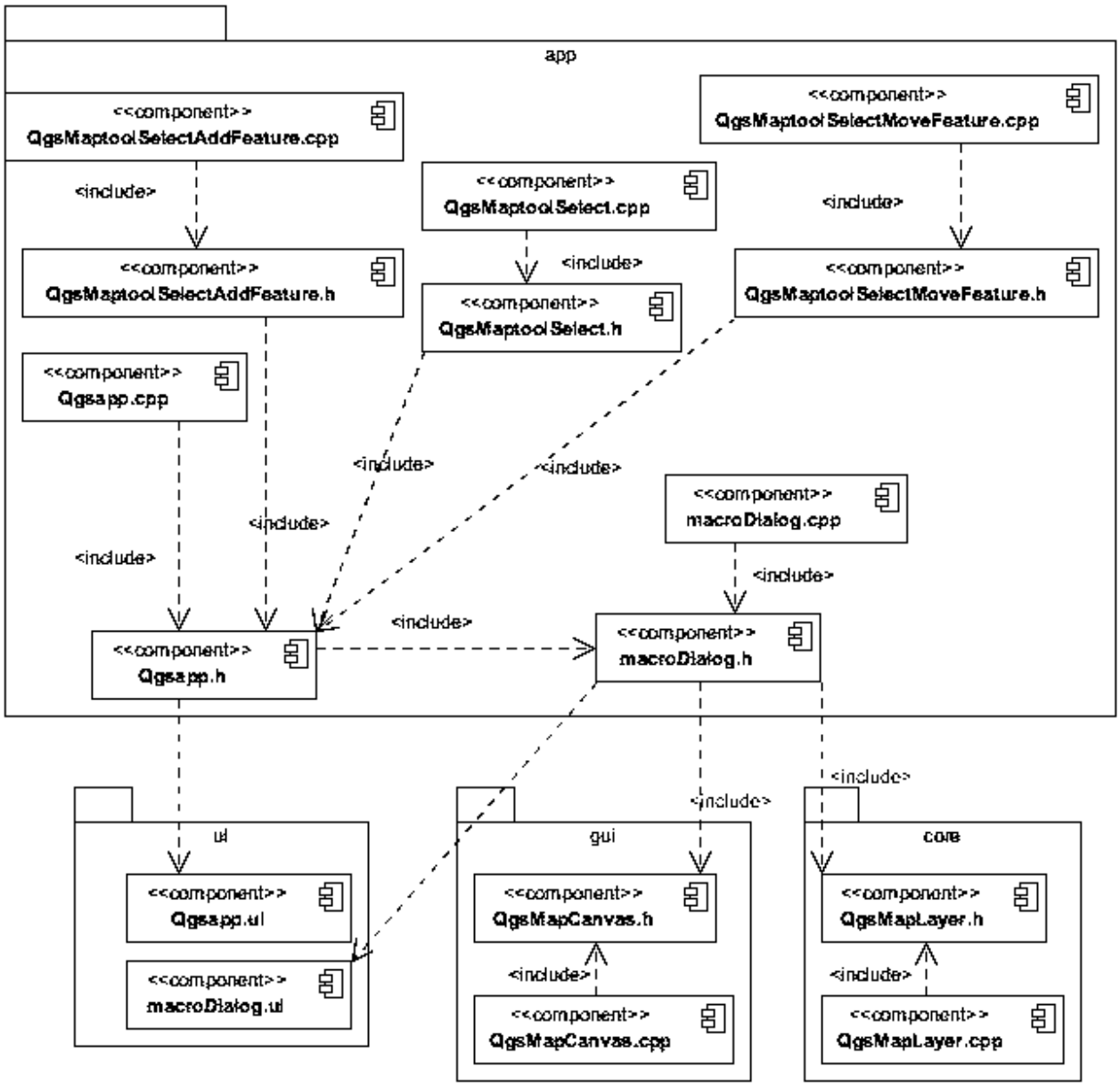

Figura 7 Diagrama de Componentes del CU "Recopilar información de edición".

Un diagrama de componentes es la representación de la forma en que los componentes físicos de un sistema serán separados, pueden ser: archivos, cabeceras, módulos, paquetes. Muestra la organización y las dependencias que existen entre un conjunto de componentes. Además, se utilizan para modelar la vista estática de un sistema.

4.12 Modelo de Despliegue

Los elementos de diseño al nivel del despliegue indican cómo se ubicará la funcionalidad y los subsistemas del entorno computacional físico que soportará al software. **(Pressman, 2005)**

Es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo, es decir, establece una correspondencia entre las arquitecturas de software y la del sistema. Cada nodo representa un recurso de cómputo, normalmente, un procesador o un dispositivo hardware similar. Los nodos poseen relaciones que representan medios de comunicación entre ellos.

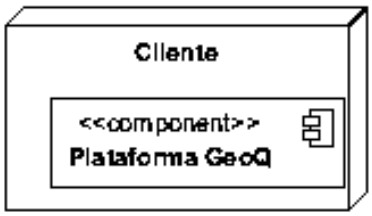

Figura 8 Diagrama de Despliegue.

4.13 Prueba del sistema propuesto

El proceso de desarrollo de software no está exento de errores; por ese motivo es necesaria la realización de pruebas, también conocidas como *test*. Estas pruebas son actividades en las cuales un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas; los resultados son observados y registrados para ser evaluados. El éxito de una prueba es el descubrimiento de un error; por tal motivo esto contribuye a mejorar la calidad del software.

Entre los diferentes tipos de pruebas se pueden destacar:

Pruebas de Validación: Son las pruebas realizadas sobre un software completamente integrado para evaluar el cumplimiento con los requerimientos especificados.

Pruebas de Sistema*:* El software ya validado se integra con el resto del sistema donde algunos tipos de pruebas a considerar son: rendimiento, robustez, seguridad, usabilidad, instalación, entre otras.

Pruebas de Caja Blanca*:* Se detectan determinados errores que con otros tipos de pruebas resultaría muy difícil posibilitando que se genere un código de mayor calidad que cumpla con los estándares.

Pruebas de Aceptación*:* Son las pruebas que realiza el cliente, determinando que el sistema cumple o no con lo especificado en los funcionales y no funcionales.

Pruebas de Caja Negra*:* Permiten obtener conjuntos de condiciones de entrada que ejecuten todos los funcionales de un programa. Este tipo de prueba no debe interpretarse como una alternativa a las

pruebas de caja blanca sino que debe interpretarse como un enfoque diferente.

4.14 Prueba de Caja Negra

Para validar de los componentes del sistema y el asegurar el cumplimiento de las funcionalidades del mismo. Se decide realizar un conjunto de pruebas de caja negra, las cuales se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que este no se ajuste a su especificación. Por ello se denominan también pruebas funcionales donde el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo realmente el módulo por dentro. Las herramientas básicas para realizarlas son: observar la funcionalidad y contrastar con la especificación.

Los casos de pruebas de la caja negra pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Tipos de errores que pretende encontrar:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores rendimiento.
- Errores de inicialización y terminación.
- Errores en estructuras de datos o en acceso a base de datos externas.

Clasificación de las técnicas de caja negra:

A continuación se detallan las técnicas de pruebas de caja negra existentes:

Prueba de análisis de valores límites: Es una técnica de diseño de casos de prueba que completa a la partición equivalente. No selecciona cualquier elemento de una clase de equivalencia, sino que lleva a la elección de casos de pruebas en sus extremos. Además, no se centra solamente en las condiciones de entrada, también analiza los campos de salida. **(Prado, 2007)**

Como un ejemplo típico de pruebas inducidas por los datos se puede mencionar la comprobación de valores límites, las situaciones de excepciones que puedan ocurrir y el rendimiento del sistema cuando cada característica del mismo se explota al límite de su capacidad. Tiene una limitación: es

prácticamente imposible reproducir todo el espectro por la innumerable cantidad de combinaciones de entradas posibles, agravadas por el desconocimiento de la lógica interna. **(Prado, 2007)**

Para la validación del sistema se seleccionó *la técnica partición equivalente.* Este método de prueba de caja negra divide el dominio de entrada de un programa en clases de datos, a partir de las cuales deriva los casos de prueba donde cada una de estas clases de equivalencia representa a un conjunto de estados válidos o inválidos para las condiciones de entrada. **(Prado, 2007)**

Secciones a probar en el Caso de Uso.

Diseño de Casos de Prueba: Recopilar información de edición

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
1. Recopilar información de edición.	EC 1.1: Recopilar información de edición con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El sistema brinda las opciones de Seleccionar, Mover objeto, Agregar objeto, Eliminar objeto, Copiar objeto, Cortar objeto, Pegar. El usuario comienza a realizar las acciones. El sistema responde en correspondencia a lo que el usuario seleccionó y lo muestra en el panel de control las acciones realizadas a la capa. El usuario presiona el botón detener el sistema para de guardar las acciones y habilita el recordar.	Seleccionar una cartografía o un conjunto de capas /Ver/ Macro Recording /seleccionar “Iniciar”.
	EC 1.2: Seleccionar con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El usuario selecciona la opción “Seleccionar”, seleccionando el área deseada, e incorpora la acción en el panel de control.	Seleccionar una cartografía o un conjunto de capas /Ver/ Macro Recording /seleccionar “Iniciar” /”Seleccionar”.

	EC 1.3: Eliminar con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El usuario selecciona la opción “Eliminar”, donde el sistema muestra una ventana emergente para confirmar la eliminación, el usuario selecciona la opción “Aceptar” y el sistema muestra la eliminación e incorpora la acción en el panel de control.	Seleccionar una cartografía o un conjunto de capas /Ver/ Macro Recording /seleccionar “Iniciar” /”Eliminar”/”Aceptar”
	EC 1.4: No realiza la operación de “Eliminar”.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El usuario selecciona la opción “Eliminar”, donde el sistema muestra una ventana emergente para confirmar la eliminación, el usuario selecciona la opción “Cancelar” y el sistema no realiza cambios en el panel de control.	Seleccionar una cartografía o un conjunto de capas /Ver/ Macro Recording /seleccionar “Iniciar” /”Eliminar”/”Cancelar”
	EC 1.5: Mover objeto con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El usuario selecciona la opción “Mover objeto”, el usuario realiza el movimiento de los objetos seleccionados al lugar deseado y finalmente incorpora la acción en el panel de control.	Seleccionar Selecciono una cartografía o un conjunto de capas /Ver/ Macro Recording /“Iniciar”/”Mover objeto”.

	EC 1.6: Copiar objeto con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El usuario selecciona la opción “Copiar objeto”, se incorpora la acción en el panel de control.	Seleccionar una cartografía o un conjunto de capas /Ver/ Macro Recording /seleccionar “Iniciar” /”Copiar objeto”.
	EC 1.7: Cortar objeto con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El usuario selecciona la opción “Cortar objeto”, se incorpora la acción en el panel de control.	Seleccionar una cartografía o un conjunto de capas /Ver/ Macro Recording /seleccionar “Iniciar” /”Cortar objeto”.
	EC 1.8: Pegar objeto con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El usuario selecciona la opción “Pegar objeto”, se incorpora la acción en el panel de control.	Seleccionar una cartografía o un conjunto de capas /Ver/ Macro Recording /seleccionar “Iniciar” /”Pegar objeto”.
	EC 1.9: Agregar objeto con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El usuario selecciona la opción “Agregar objeto”, el usuario agrega los objetos en el lugar deseado e incorpora la acción en el panel de control.	Seleccionar una cartografía o un conjunto de capas /Ver/ Macro Recording /seleccionar “Iniciar” /”Agregar objeto”.

Tabla 2 Secciones a probar en el CU “Recopilar información de edición”.

Descripción de variable.

No	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	undoView	Panel de Información	No	Este campo permite mostrar las acciones realizadas por el usuario.

Tabla 3 Descripción de variable del CU "Recopilar información de edición".

Matriz de datos

SC 1: Recopilar información de edición

ID del escenario	Escenario	Variable 1	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Recopilar información de edición con éxito.	undoView	El sistema para de guardar las acciones y habilita el botón iniciar y recordar, el panel de control muestra las acciones realizadas al objeto.	Satisfactorio.
EC 1.2	Seleccionar con éxito.	undoView	El sistema incorpora la acción en el panel de control.	Satisfactorio.
EC 1.3	Eliminar con éxito.	undoView	El sistema incorpora la acción en el panel de control.	Satisfactorio.
EC 1.4	No realiza la operación de "Eliminar"	undoView	El sistema no realiza cambios en el panel de control.	Satisfactorio.
EC 1.5	Mover objeto con éxito.	undoView	El sistema incorpora la acción en el panel de control.	Satisfactorio.

Tabla 4 Matriz de datos SC: "Recopilar información de edición".

4.15 Conclusiones Parciales

En este capítulo se desarrollaron los principales artefactos y modelos para llevar a cabo el proceso de implementación de vital importancia en la construcción del sistema. Teniendo como resultado una

herramienta capaz de automatizar las acciones del usuario, cumpliendo los requerimientos funcionales y no funcionales planteados al inicio de la propuesta de solución. Se realizaron las pruebas, con el que se pretende probar los elementos que componen la interfaz, para la validación de la solución propuesta; llegando a ser satisfactoria por lo que de esta manera se tienen las bases en la que se apoya la construcción del sistema.

CONCLUSIONES GENERALES

Después del total cumplimiento de las tareas trazadas durante la investigación, se desarrolló la Automatización de Acciones de Usuario. Obteniendo estos resultados se arribó a un conjunto de conclusiones que a continuación se relacionan:

- La solución propuesta constituye la primera versión del sistema capaz de repetir funciones en el SIG de escritorio GeoQ.
- Se tomaron herramientas, framework y tecnologías que responden a criterios de selección de tecnologías libres y multiplataforma, de acuerdo con las políticas que impulsa la universidad y el país en sentido general.
- Todos los requisitos funcionales previstos fueron implementados satisfactoriamente, teniendo en cuenta las restricciones no funcionales especificadas a lo largo de la investigación.
- Los diferentes modelos que se utilizaron en la investigación resultaron de vital importancia para el proceso de construcción de automatización de acciones. Posibilitaron la realización de la solución y una fuerte documentación para posibles versiones.
- La herramienta desarrollada es fácil de utilizar, con un diseño sencillo, puede ser ejecutado en computadoras con bajas prestaciones y desde cualquier sistema operativo.
- El diseño de las pruebas de caja negra permitió validar el cumplimiento de los requisitos funcionales capturados y comprobar que el sistema realiza las acciones que debe realizar en el momento esperado.
- La puesta en explotación de la aplicación desarrollada proporcionará mayor calidad y rapidez, facilitando la corrección de posibles errores que pueden introducirse de forma manual por el usuario.

RECOMENDACIONES

En correspondencia con los resultados obtenidos y la experiencia acumulada a lo largo de todo el proceso investigativo se proponen las siguientes recomendaciones:

- Incluir en la Ayuda del sistema la nueva herramienta creada para la capacitación de cada usuario nuevo que deba utilizar el sistema para el desarrollo de sus actividades.

- Continuar con el desarrollo de la herramienta, con la identificación de nuevos requisitos funcionales, en aras de ampliar las posibilidades de trabajo del sistema desarrollado.

- Seguir en el trabajo con las funcionalidades de Hacer/Rehacer como ayuda al usuario para facilitarle el trabajo con la plataforma GeoQ.

REFERENCIAS BIBLIOGRAFICAS

Académico, Sitio Web del. 2011. Sitio Web del Académico. *Sitio Web del Académico.* [Online] 2011. http://grupo.unavirtual.una.ac.cr/mahara/.

Betancourt, Liudmila. 2011. *Diseño y aplicación de pruebas.* La Habana : s.n., 2011.

Booch, G, Rumbaugh, J and Jacobson, I. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

Buschmann, F. (1996). *Pattern Oriented Software Architecture.*

Camacho, E. (2004). *Arquitectura de Software.*

Carrillo Pérez, Isaías, Pérez González, Rodrigo and Rodríguez Martín, David Aureliano. 2008. *Metodología de Desarrollo de Software.* 2008.

Clements, Paul. 1996. *Coming Attractions in Software Architecture.* 1996.

Comunicaciones, Grupo de Tecnología de la Informática y las. 2011. Grupo de Tecnología de la Informática y las Comunicaciones. [Online] 2011. [Cited: 11 5, 2011.] http://www.gtic.ssr.upm.es/demo/curtic/1tl101.htm.

Comunicaciones, TICS - Tecnología de la Informática y las. 2011. TICS - Tecnología de la Informática y las Comunicaciones. [Online] 2011. [Cited: 11 5, 2011.] http://tics.org.ar/index.php?option=com_content&task=view&id=13&Itemid=28.

Cueva Lovelle, J. M. (1998). CONCEPTOS BÁSICOS DE PROCESADORES DE LENGUAJE. España: SERVITEC.

Departamento de Geografía Univesidad de Alcalá. (2011). Retrieved from Departamento de Geografía Univesidad de Alcalá: http://www.geogra.uah.es/gisweb/1modulosespanyol/IntroduccionSIG/GISModule/GIST_Raster.htm

Fuentes, Lidia, Troya, Jose M and Vallecillo, Antonio. *Desarrollo de Software Basado en Componentes.* Málaga : s.n.

GNU. 2011. GNU. *GNU.* [Online] 12 2011. http://www.gnu.org/home.es.html.

Grupo de Geomorfología, Hidrogeología y Medio Ambiente. 2011. Grupo de Geomorfología, Hidrogeología y Medio Ambiente. [Online] 2011. [Cited: 11 6, 2011.] http://ggyma.geo.ucm.es/docencia/documentos/informatica/Informatica_9.pdf.

gvSIG, Asociación. 2011. Asociación gvSIG. [Online] 2011. [Cited: 11 18, 2011.] http://www.gvsig.com/products/sextante?set_language=en.

International, Visual Paradigm. 1999. Visual Paradigm. *Visual Paradigm.* [Online] 1999. [Cited: 1 28, 2012.] http://www.visual-paradigm.com/product/vpuml/communityedition.jsp.

Jacobson, Ivar. 2000. *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación, 2000. 8478290362.

Jalón, Javier García de. 1998. *Aprenda C++ como si estuviera en primero.* Navarra : s.n., 1998.

Kruchten, P. 1999. *The Rational Unified Process.* 1999.

Larman. 1999. *UML y Patrones: Introducción al Análisis y Diseño. Orientado a Objetos.* 1999.

Martínez, Alejandro and Martínez, Raúl. 2004. *Guía a Rational Unified Process.* Castilla : s.n., 2004.

2003. Practical GIS Analysis. [book auth.] David L. Verbyla. *Practical GIS Analysis.* 2003.

Prado, Elena Raja. 2007. *Actas de Talleres de Ingeniería del Software.* 2007. 1.

—. 2007. sistedes. [Online] 2007. http://www.sistedes.es/TJISBD/Vol-1/No-4/articles/pris-07-raja-ctps.pdf.

Pressman, Roger S. 2005. *Ingeniería de Software un Enfoque Práctico.* 2005.

Rational Rose Enterprise. 2007. Grupo Soluciones Innova. *Grupo Soluciones Innova.* [Online] 2007. [Cited: 1 28, 2012.] http://www.rational.com.ar/herramientas/roseenterprise.html.

Scribd. 2011. Scribd. [Online] 2011. [Cited: 11 6, 2011.] http://es.scribd.com/doc/14202815/Conceptos-de-SIG-y-Geoprocesamiento.

Sebastiá, Laura. 2011. *Apuntes de Bases de Datos Cartográficas.* Valencia : Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y Computación, 2011.

Sommerville, Ian. 2005. *Ingeniería de Software Séptima Edición .* 2005.

Tecnológico. 2012. Tecnológico. [Online] 4 15, 2012. http://www.mitecnologico.com/Main/DefinicionAutomatizacion.

Tobarra Narro, Manuel. 2003. *CMM y RUP: Una perspectiva común.* Albacete : s.n., 2003.

univalle. 2011. univalle. [Online] 12 28, 2011. http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/ExpoCommDarTatXim.pdf.

BIBLIOGRAFIA CONSULTADA

Académico, Sitio Web del. 2011. Sitio Web del Académico. *Sitio Web del Académico.* [Online] 2011. http://grupo.unavirtual.una.ac.cr/mahara/.

Booch, G, Rumbaugh, J and Jacobson, I. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

Buschmann, F. (1996). *Pattern Oriented Software Architecture.*

Camacho, E. (2004). *Arquitectura de Software.*

Carrillo Pérez, Isaías, Pérez González, Rodrigo and Rodríguez Martín, David Aureliano. 2008. *Metodología de Desarrollo de Software.* 2008.

Clements, Paul. 1996. *Coming Attractions in Software Architecture.* 1996.

Comunicaciones, Grupo de Tecnología de la Informática y las. 2011. Grupo de Tecnología de la Informática y las Comunicaciones. [Online] 2011. [Cited: 11 5, 2011.] http://www.gtic.ssr.upm.es/demo/curtic/1tl101.htm.

Comunicaciones, TICS - Tecnología de la Informática y las. 2011. TICS - Tecnología de la Informática y las Comunicaciones. [Online] 2011. [Cited: 11 5, 2011.] http://tics.org.ar/index.php?option=com_content&task=view&id=13&Itemid=28.

Cueva Lovelle, J. M. (1998). CONCEPTOS BÁSICOS DE PROCESADORES DE LENGUAJE. España: SERVITEC.

Departamento de Geografía Univesidad de Alcalá. (2011). Retrieved from Departamento de Geografía Univesidad de Alcalá: http://www.geogra.uah.es/gisweb/1modulosespanyol/IntroduccionSIG/GISModule/GIST_Raster.htm

GNU. 2011. GNU. *GNU.* [Online] 12 2011. http://www.gnu.org/home.es.html.

Grupo de Geomorfología, Hidrogeología y Medio Ambiente. 2011. Grupo de Geomorfología, Hidrogeología y Medio Ambiente. [Online] 2011. [Cited: 11 6, 2011.] http://ggyma.geo.ucm.es/docencia/documentos/informatica/Informatica_9.pdf.

gvSIG, Asociación. 2011. Asociación gvSIG. [Online] 2011. [Cited: 11 18, 2011.] http://www.gvsig.com/products/sextante?set_language=en.

International, Visual Paradigm. 1999. Visual Paradigm. *Visual Paradigm.* [Online] 1999. [Cited: 1 28, 2012.] http://www.visual-paradigm.com/product/vpuml/communityedition.jsp.

Jacobson, Ivar. 2000. *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación, 2000. 8478290362.

Jalón, Javier García de. 1998. *Aprenda C++ como si estuviera en primero.* Navarra : s.n., 1998.

Larman. 1999. *UML y Patrones: Introducción al Análisis y Diseño. Orientado a Objetos.* 1999.

Martínez, Alejandro and Martínez, Raúl. 2004. *Guía a Rational Unified Process.* Castilla : s.n., 2004.

2003. Practical GIS Analysis. [book auth.] David L. Verbyla. *Practical GIS Analysis.* 2003.

Prado, Elena Raja. 2007. *Actas de Talleres de Ingeniería del Software.* 2007. 1.

Pressman, Roger S. 2005. *Ingeniería de Software un Enfoque Práctico.* 2005.

Scribd. 2011. Scribd. [Online] 2011. [Cited: 11 6, 2011.] http://es.scribd.com/doc/14202815/Conceptos-de-SIG-y-Geoprocesamiento.

Tecnológico. 2012. Tecnológico. [Online] 4 15, 2012. http://www.mitecnologico.com/Main/DefinicionAutomatizacion.

univalle. 2011. univalle. [Online] 12 28, 2011. http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/ExpoCommDarTatXim.pdf.

ANEXOS

Anexos 1: Descripción textual de los Casos de Uso del sistema.

Caso de Uso:	Recopilar información de navegación
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario desea guardar acciones. El caso de uso termina cuando se guardan acciones.
Referencias	RF1
Prioridad	Alta
Precondiciones	Debe estar cargada alguna cartografía y estar activado la opción Macro Recording.

Flujo de eventos		
Recopilar información de navegación		
Actor		**Sistema**
	1- El usuario presiona el botón iniciar.	2- El sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El sistema brinda las opciones de Pan, Zoom.
	3- El usuario comienza a realizar las acciones. Las acciones a realizar son: - Paneo - Zoom	4- El sistema responde en correspondencia a lo que el usuario seleccionó: -Si el usuario selecciona la opción de Paneo. Ver sección Paneo. -Si el usuario selecciona la opción de Zoom. Ver sección Zoom.
Sección : “Paneo”		
	Actor	**Sistema**
	5- El usuario realiza el paneo deseado	6- El sistema incorpora la acción en el panel de control.
Prototipo de Interfaz		

Sección : “Zoom”		
	Actor	**Sistema**
	5- El usuario realiza el zoom deseado	6- El sistema incorpora la acción en el panel de control.

Prototipo de Interfaz

Postcondiciones	Se almacenó la información de las acciones realizadas por el usuario en el panel de control.

Tabla 5 Descripción del CU “Recopilar información de navegación”.

Caso de Uso:	Recopilar información de tematización
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario desea guardar acciones. El caso de uso termina cuando se guardan acciones.
Referencias	RF2

Prioridad	Alta
Precondiciones	Debe estar cargada alguna cartografía y estar activado la opción Macro Recording.

Flujo de eventos		
Recopilar información de tematización		
Actor		**Actor**
	1- El usuario presiona el botón iniciar.	2- Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas a la capa. El sistema brinda las opciones de Estilo línea, Grosor, Color, Relleno.
	3- El usuario selecciona la ventana "Propiedades de la capa"	
	4- El usuario realiza las acciones y presiona "Aceptar"	5- El sistema incorpora la acción en el panel de control

Prototipo de Interfaz

Flujo alterno		
	Actor	**Sistema**
	4.1- El usuario realiza las acciones y preciona "Cancelar"	5.1- El sistema no incorpora la acción en el panel de control.

Prototipo de Interfaz	
Postcondiciones	Se almacenó la información de las acciones realizadas por el usuario en el panel de control.

Tabla 6 Descripción del CU "Recopilar información de tematización".

Caso de Uso:	Ejecutar lista de acciones	
Actores	Usuario	
Resumen	El caso de uso se comienza cuando halla acciones almacenada. El caso de uso termina cuando el usuario ejecute las acciones a la capa deseada.	
Referencias	RF4	
Prioridad	Alta	
Precondiciones	Debe existir información en el panel de control.	
Flujo de eventos		
Ejecutar lista de acciones		
Actor		**Sistema**
	1- El usuario selecciona la capa.	
	2- El usuario presiona el botón recordar.	3- El sistema realiza las acciones mostrado en el panel de control a la capa.

Prototipo de Interfaz	
Postcondiciones	Se realizó cambios en la capa.

Tabla 7 Descripción del CU “Ejecutar lista de acciones”.

Anexos 2: Diagramas de Clases del Diseño

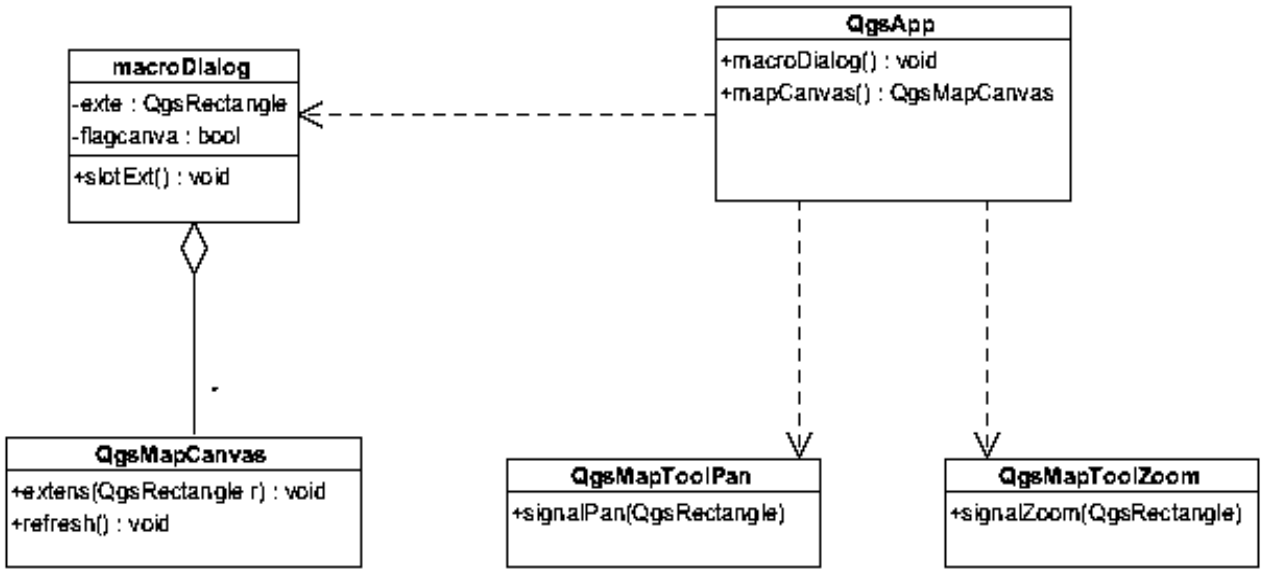

Figura 9 Diagrama de Clases del Diseño del CU "Recopilar información de navegación".

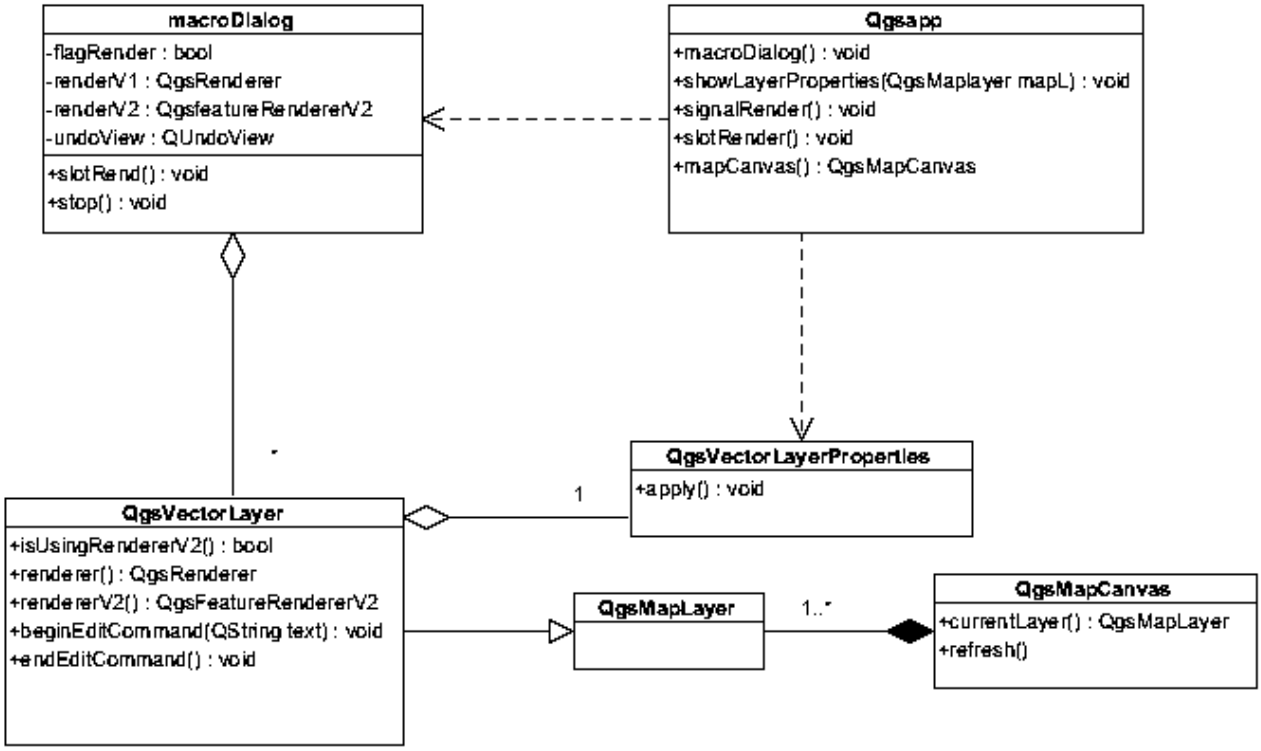

Figura 10 Diagrama de Clases del Diseño del CU "Recopilar información de tematización".

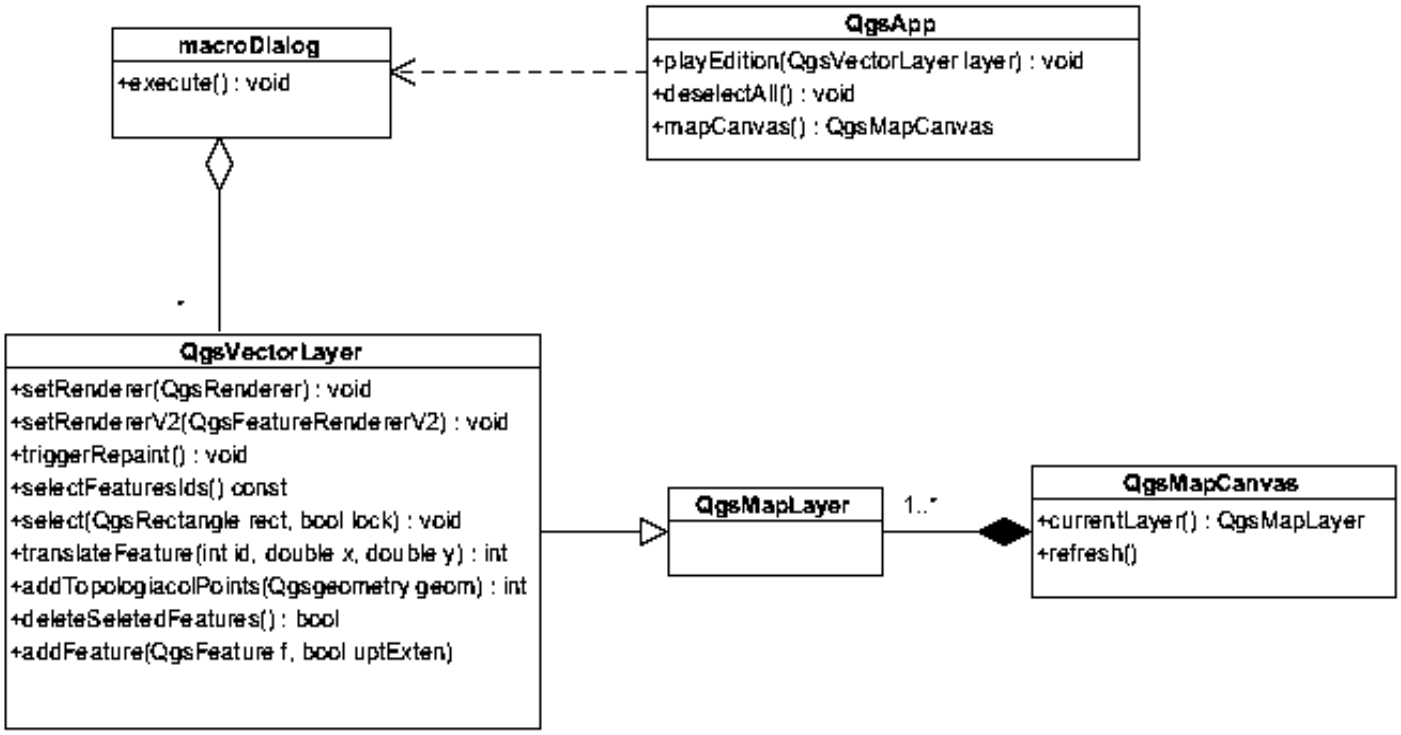

Figura 11 Diagrama de Clases del Diseño del CU "Ejecutar lista de acciones".

Anexos 3: Casos de Prueba

Secciones a probar en el Caso de Uso.

Diseño de Casos de Prueba: Recopilar información de navegación

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
1. Recopilar información de navegación	EC 1.1: Recopilar información de navegación con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El sistema brinda las opciones de Paneo y Zoom. El usuario comienza a realizar las acciones. El sistema responde en correspondencia a lo que el usuario seleccionó y lo muestra en el panel de control las acciones realizadas a la capa. El usuario presiona el botón detener el sistema para de guardar las acciones y habilita el recordar.	Seleccionar una cartografía o un conjunto de capas/Ver / Macro Recording /Seleccionar “Iniciar”.
	EC 1.2: Paneo con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El usuario selecciona la opción “Paneo”, seleccionando el área deseada, e incorpora la acción en el panel de control.	Seleccionar una cartografía o un conjunto de capas/Ver / Macro Recording /Seleccionar “Iniciar”/”Paneo”.
	EC 1.3: Zoom con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas al objeto. El usuario selecciona la opción “Zoom”, seleccionando el área deseada, e incorpora la acción en el panel de control.	Seleccionar una cartografía o un conjunto de capas/Ver / Macro Recording /Seleccionar “Iniciar”/”Zoom”.

Tabla 8 Secciones a probar en el CU “Recopilar información de navegación”.

Descripción de variable.

No	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	undoView	Panel de Información	No	Este campo permite mostrar las acciones realizadas por el usuario.

Tabla 9 Descripción de variable del CU "Recopilar información de navegación".

Matriz de datos

SC 1: Recopilar información de navegación

ID del escenario	Escenario	Variable 1	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Recopilar información de navegación con éxito.	undoView	El sistema para de guardar las acciones y habilita el botón iniciar y recordar, donde el panel de control muestra las acciones realizadas al objeto.	Satisfactorio.
EC 1.2	Paneo con éxito.	undoView	El sistema para de guardar las acciones y habilita el botón iniciar y recordar, donde el panel de control muestra las acciones realizadas al objeto.	Satisfactorio.
EC 1.3	Zoom con éxito.	undoView	El sistema para de guardar las acciones y habilita el botón iniciar y recordar, donde el panel de control muestra las acciones realizadas al objeto.	Satisfactorio.

Tabla 10 Matriz de datos SC: "Recopilar información de navegación".

Secciones a probar en el Caso de Uso.

Diseño de Casos de Prueba: Recopilar información de tematización

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
1. Recopilar información de tematización.	EC 1.1: Recopilar información de tematización con éxito.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas a la capa, el usuario selecciona la ventana “Propiedades de la capa”. El usuario realiza las acciones y presiona “Aceptar” en la ventana. El sistema incorpora la acción en el panel de control las acciones realizadas a la capa, una vez esto el usuario presiona el botón “Detener” y finalmente el sistema para de guardar las acciones y habilita el botón recordar.	Seleccionar una cartografía o un conjunto de capas/Ver / Macro Recording /seleccionar “Iniciar”/“Propiedades de la capa”/“Aceptar”
	EC 1.2: No realizar la operación de Recopilar información de tematización.	Una vez que el usuario presiona el botón iniciar, el sistema se prepara para guardar, habilita el botón detener y un panel de control para mostrar las acciones realizadas a la capa, el usuario selecciona la ventana “Propiedades de la capa”. El usuario presiona “Cancelar” en la ventana y el sistema no incorpora la acción en el panel de control.	Seleccionar una cartografía o un conjunto de capas/Ver/ Macro Recording/seleccionar “Iniciar”/“Propiedades de la capa”/“Cancelar”

Tabla 11 Secciones a probar en el CU “Recopilar información de tematización”.

Descripción de variable.

No	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	undoView	Panel de Información	No	Este campo permite mostrar las acciones realizadas por el usuario.

Tabla 12 Descripción de variable del CU “Recopilar información de tematización”.

Matriz de datos

SC 1: Recopilar información de tematización

ID del escenario	Escenario	Variable 1	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Recopilar información de tematización con éxito.	undoView	El sistema para de guardar las acciones y habilita el botón iniciar y recordar, donde el panel de control muestra las acciones realizadas al objeto.	Satisfactorio
EC 1.2	No realizar la operación de Recopilar información de tematización.	undoView	El sistema para de guardar las acciones y habilita el botón iniciar y recordar, donde el panel de control muestra las acciones realizadas al objeto.	Satisfactorio

Tabla 13 Matriz de datos SC: "Recopilar información de tematización".

GLOSARIO DE TERMINOS

CASE: (*Computer Aided Software Engineering*).Constituye una herramienta que ayuda al ingeniero de software a desarrollar y mantener software.

Datos Espaciales: Los datos geográficos se definen como cualquier información sobre objetos o fenómenos que tengan una ubicación relativa con respecto a la superficie de la Tierra.

Geoespacial: Es la información con componente espacial en el sentido geográfico.

Georreferenciación: La georreferenciación para un determinado sistema de coordenadas no es más que la posición con la que se define la localización de un objeto espacial, (esto puede ser a través de vectores, puntos, aéreas).

Multiplataforma: Término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

Ráster: Es un método para el almacenamiento, el procesado y la visualización de datos geográficos. Cada superficie a representar se divide en filas y columnas, formando una malla o rejilla regular. Cada celda ha de ser rectangular, aunque no necesariamente cuadrada. Cada celda de la rejilla guarda tanto las coordenadas de la localización como el valor temático. La localización de cada celda es implícita, dependiendo directamente del orden que ocupa en la rejilla, a diferencia de la estructura vectorial en la que se almacena de forma explícita la topología. Las áreas que contienen idéntico atributo temático son reconocidas como tal, aunque las estructuras ráster no identifican los límites de esas áreas como polígonos en sí. **(Departamento de Geografía Univesidad de Alcalá, 2011)**

Software: “Producto que los ingenieros de software construyen y después mantienen en el largo plazo. Incluyen los programas que se ejecutan dentro de una computadora de cualquier tamaño y arquitectura, el contenido que se presenta conforme los programas y los documentos, tanto físicos como virtuales, que engloban todas las formas de medios electrónicos”. **(Pressman, 2005)**

Software Libre: El software respeta la libertad de los usuarios y la comunidad. En términos generales, los usuarios tienen la libertad de copiar, distribuir, estudiar, modificar y mejorar el software. Con estas libertades, los usuarios (tanto individualmente como en forma colectiva) controlan el programa y lo que hace. **(GNU, 2011)**