UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Sistema para la georreferenciación y análisis de la distribución espacial de los tumores malignos en Cuba

Tesis presentada para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Raniel Gé Vaillant

Tutores: Ing. Liset González Polanco

Ing. Yadian Guillermo Pérez Betancourt

Ing. Jorge Jesús Hidalgo Ruíz

La Habana, 23 de junio de 2016

Una importante especie biológica está en riesgo de desaparecer por la rápida y progresiva liquidación de sus condiciones naturales de vida: el hombre.

Fidel Castro Ruz

Declaración de autoría

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

Para que así conste se firma la presente a los 23 días del mes de Junio del año 2016.

Raniel Gé Vaillant

Ing. Liset González Polanco

Ing. Yadian Guillermo Pérez Betancourt

Ing. Jorge Jesús Hidalgo Ruíz

AGRADECIMIENTOS

Agradezco a la Revolución Cubana por brindarme la posibilidad de formarme como Ingeniero.

A mi papá, por ser el guía de toda mi vida; aunque ya hace más de 4 años que no está físicamente conmigo, pero lo siento presente gracias por ser padre, hermano, amigo, mi todo.

A mis dos madre, a ti mi nana (Aliannis) por todo el tiempo que me has dedicado y a ti mi mámi por todo tu amor, preocupación, dedicación, cariño y ser la mejor madre del mundo, las amo.

A mi familia, por todo su apoyo, en especial a tí mi flaca (Yarisleydis) por un día comentarme de esta maravillosa universidad.

A ti mi hermano y mi tía sabia que no me iban a fallar, así como los que están en casa.

A mis abuelos, así como a tío Roberto, por todo su amor en vida siempre los tengo presente.

A mis tutores, por su apoyo, confianza y todo el tiempo que me han dedicado.

A Yadian, por no solo ser mi tutor en esta tesis, sino también ser mi hermano, amigo y ser el junto a Liset las personas que me recibieron al entrar a la UCI hace casi 7 años ya.

A el tribunal por sus recomendaciones y sugerencias en este proceso

A mi familia en la UCI, Miguel(Mi papa), Leydis(Mi mama), Reinier(Mi abu) el cual esta lejos de mi pero siempre esta preocupado por su nieto, mis hermanos (Leandro, Yoendrys, Eliannis, Aliankis, Yoan, Josue, Ariel, Carlos, Viset, Ali, Yazmin), mis niñas (Sheila y Dayana).

A todos los profesores de la mejor facultad de la UCI

A mis compañeros de aula y apartamento por soportarme en este tiempo.

A mis entrenadores (Joel, Prisco), a la gente del gym y los compañeros de equipo de la facultad y UCI de atletismo.

A las tías del comedor, docente y apartamento.

DEDICATORIA

A mi papa Ricardo Gé Berroa.

A mi mama Cruz Maria Vaillant Derribal.

A mi nana Aliannis Reyes Vaillant.

RESUMEN

Los tumores malignos están identificados como un problema de salud a nivel mundial debido a las altas tasas de incidencia, la mortalidad registrada y a los problemas de orden familiar, laboral y económico que generan. Varias investigaciones refieren una fuerte relación entre los factores de riesgos y el espacio. Si bien el espacio es una componente importante para conformar hipótesis referidas a su etiología, no siempre se le da la importancia requerida debido a limitaciones que presentan las herramientas existentes. El objetivo del presente trabajo es desarrollar una propuesta para la georreferenciación y análisis de la distribución espacial de los tumores malignos sobre el Sistema de Información Geográfica QGis para incorporar el espacio en estos estudios. Se obtuvo como resultado un sistema que permite realizar la georreferenciación y el análisis espacial de los casos; la solución tiene su base en las técnicas de agrupamiento y contribuye a un mejor entendimiento de las relaciones entre el espacio, las personas y los factores de riesgos asociados a los tumores malignos. Para valorar las potencialidades de la solución se realizó un caso de estudio con la información de la mortalidad registrada en el anuario estadístico del 2014.

Palabras clave: análisis espacial, tumores malignos, georreferenciación, algoritmo de agrupamiento espacial.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los tumores malignos son desde el punto de vista médico una enfermedad difícil de abordar por la multicausalidad de su origen, la complejidad de sus mecanismos patogénicos y la variedad de formas y tipos que pueden originarse en el ser humano. Están identificados como un problema de salud a nivel mundial [1], sustentado en las altas tasas de incidencia y mortalidad registradas, y en los problemas de orden familiar, laboral y económico que generan.

Se estima que para el año 2020 más de 1,4 millones de personas morirán por esta causa, debido a los cambios demográficos y una mayor exposición a los factores de riesgo [2]. Este elemento ha potenciado que países del primer mundo entre lo que se encuentran Canadá, Estados Unidos, España, Inglaterra, Francia, Italia y Japón diseñen sus estrategias, para dar prioridad a los trabajos asociados a su prevención y análisis. Estas estrategias no han tenido un carácter regional o global pues han estado orientadas a zonas de su territorio [1]. Cuba, a pesar de ser un país subdesarrollado ha mostrado grandes avances en los trabajos asociados a la prevención y análisis de esta enfermedad.

En Cuba el cáncer o *"asesino silencioso"* como también se conoce, desde el año 1958 constituye la segunda causa de muerte, solo superada por las enfermedades del corazón. En el año 2010, el número de fallecidos por tumores malignos fue de 21 035 para una tasa de 198,4 por cada 100000 habitantes. El 26,19 % de todas las muertes producidas es atribuible al cáncer con un incremento de 4 % en los últimos 30 años, según refieren las estadísticas del Ministerio de Salud de Cuba sobre el tema [3]. Esta cifra está relacionada con que la mortalidad por esta enfermedad constituye la primera causa de defunción entre las edades de 50 a 64 años y la segunda de 65 años en lo adelante y al aumento de la esperanza de vida al nacer de los cubanos [4].

Cada año se diagnostican más de 27 000 nuevos enfermos [5] y se registran más de 18 000 fallecidos [4]. Desde 1990 al 2005, la mortalidad por cáncer en Cuba tuvo un incremento del

2 % anual y desde el año 2006 al 2012, la mortalidad por los tumores malignos ha seguido en aumento con una tasa de incremento del 3 %. Desde el año 2012 los tumores malignos constituyen la primera causa de muerte, con una tasa en 2012 de 201,4 por cada 100000 habitantes, con tendencia al aumento. En el año 2014 la tasa registrada fue de 212,6, lo que representa un aumento 5,5 % con respecto a la tasa del 2012.

El aumento de la mortalidad por tumores malignos apunta a un problema más serio en el futuro de Cuba, a partir del proceso de envejecimiento de la población. Este proceso de envejecimiento implicará una mayor demanda de recursos humanos, materiales y financieros para el tratamiento de esta enfermedad, así como para la ejecución de acciones para la prevención.

Las principales herramientas de los planificadores y administradores de salud para evaluar el estado de salud en la población están orientadas a definir prioridades y asignar recursos para la vigilancia de problemas específicos en la salud[1]. En el estudio epidemiológico de cualquier enfermedad varios son los aspectos a ser tratados: su evolución, los factores o prácticas de riesgo, los diagnósticos de morbilidad o patologías asociados, su distribución espacial y las características de la población afectada [6]. Numerosas investigaciones refieren que en algunas regiones se registran casos con más frecuencia que en otras [7][8] por lo que el conocimiento de la distribución geográfica del riesgo de morir por cáncer, constituye un aporte para el establecimiento de políticas de salud.

La componente espacial ha sido empleada para identificar diferencias en la frecuencia de cáncer a nivel internacional, ya sea en términos de incidencia o en mortalidad [9][7]. Se ha observado distribuciones que han permitido plantear hipótesis etiológicas que expliquen tales variaciones. Las diferencias geográficas pueden ser interpretadas como una expresión de las características genéticas o sociales de las poblaciones más que de un factor físico-natural propio de cada región [1].

Estos elementos indican que el comportamiento en las estadísticas de la mortalidad por los tumores malignos en Cuba ameritan estudios detallados sobre el comportamiento espacial de la patología que apoyen la planificación de recursos y a la conformación de hipótesis referidas a su etiología [1].

Los estudios presentados en la literatura evidencian un profundo análisis estadístico de los casos

de mortalidad por la enfermedad en Cuba, no siendo así en el análisis espacial. En este sentido la incorporación de técnicas de análisis espacial en los estudios del comportamiento espacial del cáncer y el desarrollo de herramientas que le den soporte constituyen una necesidad de primer orden.

Los trabajos asociados a estos estudios dentro de la salud utilizan los Sistemas de Información Geográfica (**SIG**) como herramienta de soporte, sin embargo su empleo en estudios sobre distribución de cáncer se limita a llevar información a la cartografía, sin explotar en su totalidad la componente espacial de los datos. Si bien el espacio es una componente importante por su relación con el surgimiento de enfermedades, no siempre se le da la importancia requerida, en el contexto nacional motivado fundamentalmente por:

- Acceso limitado a los SIG por los costos que ellos implican
- Poco conocimiento de las herramientas
- El tiempo de formación en el área de los SIG es elevado

En Cuba se ha reconocido además que las distribuciones espaciales de los problemas de salud, han sido poco estudiadas a pesar de contarse con información de reconocida calidad.

A partir del diagnóstico preliminar de la situación actual, se identifica el siguiente **problema a resolver**: ¿cómo incorporar la componente espacial en el análisis de la distribución de tumores malignos en Cuba?

Se delimitó como **objeto de estudio** la georreferenciación y análisis de la distribución espacial de enfermedades y se formuló la siguiente **idea a defender:** un sistema informático para la georreferenciación y análisis de la distribución espacial de los tumores malignos en Cuba, facilitará la obtención de información con valor añadido para la planificación de recursos y la conformación de hipótesis referidas a su etiología.

Para dar solución al problema planteado se trazó como **objetivo general**: desarrollar un sistema informático para la georreferenciación y análisis de la distribución espacial de tumores malignos en Cuba que facilite la obtención de información de valor añadido para la planificación de recursos y la conformación de hipótesis referidas a su etiología. El **campo de acción** estará enmarcado en las técnicas y herramientas para el análisis de la distribución espacial de tumores malignos.

El objetivo general se desglosó en los siguientes **objetivos específicos**:

- Construir el marco teórico referencial de la investigación, relacionado con el estudio de la distribución espacial de enfermedades.
- Implementar un componente para Qgis que permita realizar la georreferenciación de los tumores malignos.
- Implementar un componente para Qgis que permita realizar análisis de la distribución espacial de los tumores malignos.
- Verificar la solución informática propuesta aplicando diferentes pruebas y métricas.

En la consecución de los objetivos trazados se utilizaron los siguientes **métodos**:

- Análisis-síntesis: para identificar los elementos que componen la naturaleza espacial de los problemas de salud, las técnicas de análisis y los principales enfoques, para profundizar en el estudio de cada elemento y luego sintetizarlos en la solución propuesta.
- Histórico-lógico: para analizar la evolución de los enfoques empleados en las técnicas de análisis espacial de problemas de salud, los modelos de análisis basados en SIG, así como de los algoritmos de detección de conglomerados más relevantes existentes en la literatura.
- Modelado: para el diseño de la herramienta computacional de georreferenciación y análisis de tumores malignos.

Estructura del trabajo

El trabajo está estructurado en introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y glosario de términos. A continuación se muestra una breve descripción de cada capítulo.

Capítulo 1: Fundamentos teóricos sobre la georreferenciación y análisis espacial.

En este capítulo se presentan elementos teóricos relacionados con la georreferenciación y análisis de la distribución espacial para lograr un mejor entendimiento del trabajo a desarrollar. Se hace una valoración comparativa sobre varios SIG existentes y sobre las principales herramientas que soportan la georreferenciación y análisis de la distribución espacial de eventos. Además, se realiza un estudio de la metodología, herramientas, tecnologías y lenguajes a utilizar

en el desarrollo de la solución.

Capítulo 2: Sistema para la georreferenciación y análisis de la distribución espacial de tumores malignos.

En este capítulo se realiza una descripción general de la solución propuesta, se especifican los requisitos funcionales y no funcionales que se tendrán en cuenta para su implementación, y se detallan aspectos relacionados con el diseño y la arquitectura del sistema. Se especifican los patrones del diseño aplicados, los artefactos derivados de la metodología de desarrollo de software seleccionada y se describe la etapa de implementación.

Capítulo 3: Validación de la solución propuesta.

En este capítulo se elaboran y documentan las pruebas realizadas a la solución propuesta para demostrar el correcto funcionamiento de la misma, y por último se analizan los resultados obtenidos tras la utilización del sistema en un caso de estudio.

Capítulo 1

Referentes teóricos sobre la georreferenciación y análisis espacial

1. REFERENTES TEÓRICOS SOBRE LA GEORREFERENCIACIÓN Y ANÁLISIS ESPACIAL

En este capítulo se presentan los elementos teóricos resultantes del análisis de la literatura consultada. Primeramente se analizan los principales referentes sobre la georreferenciación y análisis espacial. Se hace una valoración comparativa sobre varios SIG existentes. Además, se realiza un estudio de la metodología, herramientas, tecnologías y lenguajes a utilizar en el desarrollo de la solución.

1.1. Marco teórico

En la presente sección se muestran varias definiciones relacionadas con los elementos principales en que se enmarca la investigación.

1.1.1. Georreferenciacíon

La **georreferenciación** es el uso de coordenadas de mapa para asignar una ubicación espacial a entidades cartográficas. Todos los elementos de una capa de mapa tienen una ubicación geográfica y una extensión específicas que permiten situarlos en la superficie de la Tierra o cerca de ella. La capacidad de localizar de manera precisa las entidades geográficas es fundamental tanto en la representación cartográfica como en SIG [10] [11].
Teniendo dos métodos de representación fundamentalmente que son los siguiente:

- La **georreferenciación orbital**: en la que se modelan las fuentes de error geométrico conocidas (la curvatura terrestre, la distorsión panorámica, la rotación en superficie). Estos aplican transformaciones inversas que corrijan estos errores intrínsecos y sistemáticos de forma automatizada. Tiene la principal ventaja de que no necesita intervención humana

una vez que es implementado, pero puede dar lugar a grandes errores en las coordenadas de las imágenes de satélite si su sistema de posicionamiento no tiene la suficiente precisión (problema que ha disminuido con la llegada de los sistemas de navegación modernos)[12].

- La **georreferenciación por puntos de control**: en la que a partir de un conjunto de puntos bien identificados en la imagen y de los que se conocen sus coordenadas se calculan las funciones de transformación (lineales, cuadráticas) que mejor se ajustan a estos puntos. Para que esta georreferenciación resulte satisfactoria es necesario elegir de forma apropiada los puntos de control (en número, ubicación y distribución). Se trata, pues, de un proceso manual en el que se requiere intervención humana. Ofrece mayor exactitud cuando se trabaja en zonas donde es posible identificar bien los puntos conocidos[12].

En está investigación se utilizará este último enfoque por sus características y la relación con las fuentes primarias de información.

1.1.2. Análisis espacial

El análisis espacial constituye una metodología útil para la gestión de riesgos y vigilancia epidemiológica del entorno social [13]. Se puede definir como las técnicas y métodos que pueden ser aplicados a los datos geográficos para agregarles valor de soporte a la toma de decisiones. Tiene como objetivo revelar patrones y anomalías sobre el comportamiento espacial de variables o eventos. En otras palabras, el análisis espacial es el proceso del cual se busca separar, procesar, clasificar y presentar con criterios cartográficos alguna información. Es el estudio cuantitativo y cualitativo de aquellos fenómenos que se manifiestan en el espacio con la idea de transformar los datos iniciales en información útil [14].

1.1.2.1. Elementos asociado al análisis espacial

En los estudios de análisis espacial se identificaron varias técnicas entre las que se destacan las siguientes:

- Proximidad: Se basada en métricas de distancias.
- Agregación: Agregar características espaciales dentro de una región.

- Interpolación: Aplica modelos de interpolación para suavizar polígonos a partir de la agregación o para rellenar regiones no observadas.
- Regresión: Utiliza modelos de regresión, entre los que se pueden destacar:
  - Los modelos de autorregresión
  - Los modelos de regresión bayesiana
- Detección de conglomerados:

  Según la literatura especializada de epidemiología se denomina conglomerado o *cluster* a un exceso de casos de enfermos diagnosticados superior a lo esperado en un área geográfica determinada (conglomerado espacial), en un período de tiempo limitado (conglomerado temporal), o considerando ambos dominios (conglomerado espacio-temporal)[15].

A partir de las características de estas técnicas y los objetivos de la presente investigación, este trabajo estará enmarcado en la detección de conglomerados espaciales. Esta técnica tiene una estrecha relación con la minería de datos espaciales, particularmente en los algoritmos de agrupamiento espacial. A continuación se define el problema de agrupamiento.

Sea $O = \{O_1, \quad O_2, \quad \ldots, \quad O_n\}$ el conjunto de n objetos o análogamente (casos, patrones, n-uplas, puntos), $R = \{r_1, r_2, \ldots, r_d\}$ un conjunto de d rasgos o atributos en término de los cuales serán representados los objetos. Se entiende por una representación del *i-ésimo* O_i el d-uplo $X_i = (r_1(O_i), \ldots, r_d(O_i)) = (x_{i1}, \ldots, x_{id}) \in \Omega = \Omega_1 \times \ldots \times \Omega_d$, donde $r_j(O_i) = x_{ij} \in \Omega_j$ es el valor que toma el rasgo r_j en el objeto O_i. El conjunto Ω_j contiene los valores admisibles del rasgo r_j (dominio de definición de la variable). Se denota $\mathbf{X} = \{\mathbf{x_1}, \quad \mathbf{x_2}, \quad \ldots, \quad \mathbf{x_n}\}$ como una representación del conjunto de objetos O donde X_i es una representación del objeto O_i. La representación X puede verse como una matriz de dimensiones $n \times d$ que sirve de entrada a la mayoría de los algoritmos de agrupamiento estudiados en la literatura consultada.

Entre las representaciones de objetos se define una función de semejanza β que se aplica a cada par de objetos O_i, O_j e indica qué tan semejantes son entre sí. A partir de estos elementos, la meta de todo algoritmo de agrupamiento consiste en asignar cada objeto a un sistema finito de subconjuntos o *clusters* que usualmente no se interceptan entre sí y cuya unión es igual al conjunto de objetos con la posible excepción de *outliers*, de modo tal que objetos similares

pertenezcan al mismo *cluster*, mientras que los objetos de *clusters* diferentes sean lo menos semejantes posible.

En [16, 17] se presenta una introducción a los algoritmos de agrupamiento, además se puede consultar en [18, 19] el enfoque estadístico de las técnicas de clasificación no supervisada y en [20] aparece un resumen de diferentes técnicas de agrupamiento útiles para un mejor entendimiento.

Los algoritmos de agrupamiento han sido empleados en reconocimiento del habla, en segmentación de imágenes y visión por computador [21, 22]; en minería de datos para extraer conocimiento desde fuentes de datos, en la recuperación de información y minería de textos [23, 24, 25, 26], en minería de datos geoespaciales [27, 28, 29] y [28], en análisis de datos heterogéneos [30], en aplicaciones Web [31, 32], en biología computacional para el análisis de ADN[1] [33] por solo mencionar algunas.

Los algoritmos de agrupamiento pueden dividirse en varias categorías según el procedimiento que utilizan para agrupar los objetos[34]:

- Algoritmos jerárquicos, que pueden ser aglomerativos y divisivos.
- Métodos por partición, entre ellos: algoritmos de reubicación, agrupamientos probabilísticos, métodos de *k-medoides* y métodos k-Medias (*k-Means*).
- Algoritmos basados en densidad, entre ellos los algoritmos de agrupamiento por conectividad basados en densidad y los agrupamientos basados en funciones de densidad.
- Métodos basados en rejillas.

Los algoritmos jerárquicos, como su nombre indica, construyen una jerarquía de agrupamientos, uniendo o dividiendo los grupos de acuerdo a una cierta función de similaridad/disimilaridad entre los grupos. Los métodos de agrupamiento jerárquicos se categorizan en aglomerativos y divisivos.

Un agrupamiento aglomerativo, generalmente, comienza con grupos unitarios (*singleton clusters*) y, recursivamente, une dos o más *clusters* apropiados. Un agrupamiento divisivo, generalmente, comienza con un sólo *cluster* en el que están todos los puntos o datos y, recursivamente, divide el *cluster* más apropiado. El proceso continúa hasta que se alcanza algún

[1] Disponible en: "http://www.enciclopediasalud.com/definiciones/adn"

criterio de parada (frecuentemente el número k de *clusters*).

Entre las ventajas de los algoritmos de agrupamiento jerárquicos se puede mencionar la flexibilidad con respecto al nivel de granularidad, son fáciles de manejar y son aplicables a cualquier tipo de atributo. Entre las desventajas se encuentra la no existencia de un criterio de parada y que, luego de construidos los *cluster*, no vuelven a ser visitados para mejorarlos.

Los métodos por partición emplean diferentes técnicas de reubicación para asignar los puntos o datos a uno de los k *clusters*; algunos buscan puntos específicos para luego reubicar los restantes, otros tienen un enfoque probabilístico. A diferencia de los métodos jerárquicos tradicionales, en algunos casos, los *clusters* son revisitados después de construidos y los puntos reubicados para mejorar el agrupamiento.

Estos algoritmos muchas veces asumen un conocimiento a priori del número de *clusters* en que debe ser dividido el conjunto de datos. La idea más usada es hallar los centroides, uno para cada *cluster* y, luego, ubicar los restantes puntos en el grupo del *centroide* más cercano. Este método tiene como desventaja que falla cuando los puntos de un *cluster* están muy cerca del *centroide* de otro grupo.

Los algoritmos basados en densidad tratan de identificar *clusters* en zonas altamente pobladas. Emplean diferentes técnicas para determinar los grupos: por grafos, basadas en histogramas, kernels, aplicando la regla K vecinos más cercanos (**K-NN**) o tratando de descubrir subgrupos denso-conectados.

Los algoritmos basados en rejillas trabajan con los datos indirectamente, construyendo resúmenes de los datos sobre el subconjunto del espacio de atributos, realizan una segmentación del espacio y seleccionan segmentos apropiados.

La investigación se centra en el estudio de los algoritmos: k-Means, DBScan y propagación por afinidad, que fueron seleccionado por el uso y los resultados mostrados en la literatura especializada [35, 36]. A continuación se describen los algoritmos:

- La función criterio frecuentemente usada en técnicas de agrupamiento por partición es el error cuadrático (*squared error*), que generalmente funciona bien con *clusters* compactos y bien separados. El error cuadrático de un agrupamiento formado por k grupos se expresa mediante la ecuación:

$$SE = \sum_{j=1}^{k} \sum_{i=1}^{N_j} \left\| x_i^j - c_j \right\|^2 \tag{1.1}$$

donde x_i^j y c_j son el *i-ésimo* patrón y el *centroide* del *j-ésimo cluster* respectivamente.

El algoritmo *k-Means* [37], [38] y [39] es uno de los más simples y conocidos algoritmos de agrupamiento. Está basado en la optimización del error cuadrático, que sigue una forma fácil para dividir una base de datos dada en k grupos fijados a priori. La idea principal es definir k centroides (uno para cada grupo) y, luego, ubicar los restantes puntos en la clase de su *centroide* más cercano. El próximo paso es recalcular el *centroide* de cada *cluster* y reubicar nuevamente los puntos en cada grupo. El proceso se repite hasta que no haya cambios en la distribución de los puntos en una iteración a la siguiente.

k-Means (clásico)

Entrada:

- $X \rightarrow$ conjunto de datos
- $k \rightarrow$ número de *clusters*

1. Seleccionar aleatoriamente k centros.
2. Repetir mientras haya cambios en la composición de los grupos.
   - 2.1. Asignar los ejemplos al *cluster* con centro más cercano.
   - 2.2. Calcular los nuevos centroides de los grupos.

Algunos de los principales inconvenientes de estos esquemas son:

1. Fallan cuando los puntos de un *cluster* están muy cercanos al *centroide* de otro grupo.
2. No obtienen buenos resultados cuando los *clusters* tienen diferentes formas y tamaños.
3. Son muy susceptibles al problema de la inicialización.
4. Son muy sensibles a los *outliers* que pueden distorsionar gravemente el resultado.
5. Sólo pueden emplearse en espacios de atributos numéricos por la necesidad de calcular el punto medio.

- Los algoritmos basados en densidad localizan zonas de alta densidad separadas por regiones de baja densidad. **DBScan** (*Density Based Spatial Clustering of Aplications with*

Noise) [40], es uno de los primeros algoritmos de agrupamiento que emplea este enfoque. Comienza seleccionando un punto t arbitrario, si t es un punto central, se empieza a construir un *cluster* alrededor de él, tratando de descubrir componentes denso-conectados; si no, se visita otro objeto del conjunto de datos.

Puntos centrales (*core points*) son aquellos tales que en su vecindad de radio **Eps**[2] , hay una cantidad de puntos mayor o igual que un umbral **MinPts**[3] especificado. Un punto borde o frontera tiene menos puntos que MinPts en su vecindad, pero pertenece a la vecindad de un punto central. Un punto ruido (*noise*) es aquel que no es ni central ni borde. La figura 1.1 ilustra cada uno de esos conceptos: si MinPts es mayor o igual a 4 y menor o igual a 6, A es un punto central, B es un punto borde y C es un punto ruido.

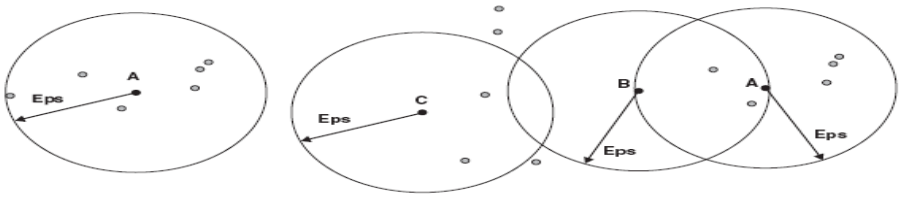

Figura 1.1: *Definiciones de punto central, borde y ruido. Tomado de [34]*

Un punto q es directamente denso-alcanzable desde otro punto t (con relación a los parámetros MinPts y Eps) si t es un punto central y q pertenece a la vecindad de t . Un punto q es denso-alcanzable desde un punto t si existe una cadena de puntos $t_0, t_1, ...t_m$, tales que t_{i-1} es directamente denso-alcanzable desde $t_i, 1 \leq i \leq m, t_0 = q$ y $t_m = t$.

En consecuencia, los puntos centrales están en regiones de alta densidad, los puntos borde en la frontera de regiones densas y los puntos ruido en regiones de baja densidad. Este algoritmo busca *cluster* comprobando la vecindad de cada punto en la base de datos y va añadiendo puntos que son denso-alcanzables desde un punto central.

[2]Radio de la vecindad de cada punto

[3]Número mínimo de puntos en una vecindad

Algoritmo DBScan

Entrada:

- $X \rightarrow$ conjunto de datos
- Eps $\rightarrow$ radio de la vecindad de cada punto
- MinPts $\rightarrow$ número mínimo de puntos en una vecindad

1. Seleccionar aleatoriamente un punto t .
2. Si t es un punto central se forma un grupo alrededor de t con todos los puntos denso-alcanzables desde t.
3. Si t es un punto borde o ruido, se visita otro punto.
4. Si todos los puntos han sido visitados, terminar; si no, volver al paso 1.

Entre sus ventajas se pueden mencionar:

1. Descubre *clusters* de formas arbitrarias.
2. Trata el ruido.
3. Es de una sola pasada.
4. Genera automáticamente el número de *clusters*.

Entre sus limitaciones:

1. Es sensible a los parámetros.
2. No es bueno para datos de alta dimensionalidad, grupos de diferentes densidades y grupos muy solapados.

En general, DBScan puede manejar *clusters* de formas y tamaños diferentes, pero tiene limitaciones cuando los *clusters* están muy solapados y en presencia de ruido.
La regla de clasificación de los K-NN ha sido extensamente empleada en métodos de clasificación supervisada [41] y [42]. En [43], se propone utilizar la regla como estrategia basada en densidad, para tratar bases de datos de alta dimensionalidad (por ejemplo, imágenes de satélites). El algoritmo DBScan comienza asignando cada punto a un *cluster* individual, luego los puntos se van asignando a uno de los grupos según la regla K-NN, terminando el proceso cuando en dos iteraciones sucesivas ninguno de los objetos cambia de grupo.

- Propagación por afinidad (de inglés *affinity propagation*) determina los *clusters* mediante el envío de mensajes entre pares de ejemplos hasta converger. De esta forma un conjunto de datos es descrito empleando un número pequeño de ejemplos, que son identificados como los más representativos del resto de los ejemplos. Toma como conjunto de datos principal las similitudes entre los datos, donde las similaridades $s(i, k)$ indican cuán adecuados son los datos k para cada punto de i. Cuando el objetivo es minimizar el error cuadrático, cada similaridad se establece como el inverso del error cuadrático.

  La propagación por afinidad parte de un conjunto de datos bidimensional, donde la distancia euclidiana se usa como medidas de similaridad. Cada punto se colorea dependiendo de la evidencia de ser el centro del cluster. Las distancias entre un punto i y un punto k son medidas mediante la fuerza por la que pueden transmitirse entre sí. La responsabilidad, $r(i, k)$ se envía entre los puntos indicando lo fuerte que es un punto en relación con otro punto ejemplar. La disponibilidad, $a(i, k)$ se envía desde los candidatos a los puntos para indicar el grado en el que estos pueden ser el centro del cluster. Después, se muestra el efecto del valor de la preferencia de entrada (común para todos los puntos de datos) en el número de ejemplares identificados (número de grupos).

  A continuación se tienen dos tipos de mensajes que se intercambian entre los datos, y cada uno de ellos tiene en cuenta un tipo distinto de competencias. Estos mensajes pueden combinar cada paso para decidir que puntos son ejemplares y así, seleccionar a cual pertenecen. En otras palabras, $r(i, k)$ y $a(i, k)$ pueden verse como cocientes de la probabilidad logarítmica. Siendo $a(i, k) = 0$, las responsabilidades se obtienen de la siguiente manera:

  $r(i, k) \leftarrow s(i, k) - \max a(i, k') + s(i, k')$, donde $k' \neq k$. Tomado de [44]

  En la primera iteración, siendo las disponibilidades cero, $r(i, k)$ se escoge como la similaridad entre el punto i y el punto k como punto ejemplar, menos la mayor de las disimilaridades entre el punto i y cada uno de los otros candidatos a ejemplar. Después, cuando los puntos se asignan a los ejemplares, sus disponibilidades serán menores que 0 tal y como se muestra a continuación .Para $k = i$, la responsabilidad $r(k, k)$ se selecciona como la preferencia donde el punto k puede escogerse como ejemplar. Esta

autoresponsabilidad refleja evidencias acumuladas de que cada punto k es un ejemplar, basado en la preferencia de entrada de cómo se ha asignado otro ejemplar.

Cuando la responsabilidad anterior permita a todos los candidatos a ejemplar competir por ser el punto central, un buen ejemplar debe seguir la siguiente forma: $a(i,k) \leftarrow \text{mín}\,0, r(k,k) + \text{máx}\,0, r(i',k)$, donde $(i' \neq i, k)$. Tomado de [45]

La disponibilidad $a(i,k)$ se establece como la autoresponsabilidad $r(k,k)$ más la suma de las responsabilidades positivas de k recibidas de cada punto. Solo las porciones positivas de las responsabilidades entrantes son añadidas porque es lo único necesario para que un ejemplar pueda explicar bien sus puntos, independientemente de que no explique bien otros puntos. Si la autoresponsabilidad $r(k,k)$ es negativa, la disponibilidad del punto k como ejemplar puede incrementarse si otros puntos tienen responsabilidades positivas para cada punto k siendo su ejemplar. Para limitar una fuerte influencia por parte de responsabilidades entrantes positivas, el total de la suma del umbral no puede ser 0. La autodisponibilidad $a(k,k)$ se actualiza de la siguiente manera:

$a(k,k) \leftarrow \text{máx}\,0, r(i',k)$, donde $(i' \neq k)$.Tomado de[45]

Este mensaje refleja la evidencia acumulada de que el punto k es el ejemplar, basado en las responsabilidades positivas enviadas al candidato ejemplar k desde otros puntos.

En cualquier momento durante la propagación por afinidad, las disponibilidades y responsabilidades pueden combinarse para identificar ejemplares. Para cada punto i, cada valor de k maximiza $a(i,k) + r(i,k)$ identificando el punto i como ejemplar si $k = i$. El procedimiento de intercambio de mensajes debe terminar cuando se alcance un número determinado de iteraciones, cuando no se produzcan cambios significativos o bien cuando las decisiones locales permanezcan constantes durante varias iteraciones consecutivas. Cuando se actualizan los mensajes, es importante comprobar la amortiguación para evitar que surjan oscilaciones. Cada mensaje se establece λ veces más $1 - \lambda$ su valor prescrito, donde el factor de amortiguamiento λ se sitúa entre 0 y 1.

La habilidad principal de la propagación por afinidad es operar sobre la base del criterio de optimización que es ajustable para análisis exploratorio de datos usando medidas de similaridad inusuales. La ventaja de utilizar este método frente a otros como el de

k-Means, radica en que puede aplicarse a problemas donde los datos no pertenecen a un espacio continuo. De hecho, se puede aplicar en problemas donde las similaridades no son simétricas $[i.e., s(i,k) \neq s(k,i)]$ y problemas donde las similaridades no satisfacen la desigualdad triangular $[i.e., s(i,k) < s(i,j) + s(j,k)]$.

1.2. Sistema de Información Geográfica

En este epígrafe se realiza un análisis crítico de las principales conceptos asociados a los sistemas de información geográfica. Los SIG poseen gran importancia tanto en la esfera social como económica. Atendiendo además que la solución que se propone en esta investigación va encaminada a este tipo de sistemas, se hace imprescindible abordar los elementos fundamentales de los mismos.

1.2.1. Definición de Sistema de Información Geográfica

Un SIG es un sistema computacional para la entrada, manejo (almacenamiento y recuperación de información), manipulación, análisis y representación de datos geográficos [46].

Los conceptos clásicos de SIG han evolucionado para destacar el papel de la diseminación de los datos como una función ineludible de los SIG en ambientes distribuidos y globales de acceso a datos y en el entorno de internet [47].

Puede definirse también como un modelo de una parte de la realidad referido a un sistema de coordenadas terrestres, construido principalmente para satisfacer la necesidad de información y ubicación geográfica del mundo. Los SIG son capaces de ubicar un objeto determinado en el espacio; encontrar dónde está un cuerpo con respecto a otro; brindar información sobre su perímetro y área; encontrar el camino mínimo de un punto a otro, así como la generación de modelos a partir de fenómenos o actuaciones simuladas [48].

Se podrían citar otras definiciones, pero en esencia pueden concretarse como: sistemas informáticos, con la capacidad de visualizar y analizar información georreferenciada de forma versátil e intuitiva, agilizando la toma de decisiones.

1.2.2. Conceptos básicos asociados a los SIGs

La **cartografía** es el arte de trazar mapas geográficos; ciencia que los estudia y se encarga de la elaboración de estos [49]. Abarca la creación y el estudio de mapas territoriales y de diferentes dimensiones lineales. También se denomina cartografía a un conjunto de documentos territoriales referidos a un ámbito concreto de estudio.

Los **datos espaciales** contienen las ubicaciones y formas de características cartográficas. Dentro de su contexto, almacenan información sobre la localización y las formas de un objeto geográfico y las relaciones entre ellos, normalmente con coordenadas y topología [50].

Un **mapa** se define como la representación convencional gráfica de fenómenos concretos o abstractos, localizados en la Tierra o en cualquier parte del Universo [50].

Un **mapa topográfico** es el que representa gráficamente los principales elementos que conforman la superficie terrestre, como vías de comunicación, entidades de población, hidrografía y relieve, con una precisión adecuada a la escala [50].

Un **mapa temático** se puede definir entonces, como aquel que está diseñado para mostrar fenómenos, eventos, características y conceptos particulares utilizando como base geográfica un mapa topográfico- [51]).

1.2.3. Panorama actual de aplicaciones SIG

Actualmente existe una gran diversidad de SIG, cada uno de ellos con numerosas alternativas, por lo que resulta complejo elegir una que se ajuste a cada necesidad. Para esto es necesario tener una visión global de las características que los diferencian. A continuación se realiza un breve análisis de algunas de las principales aplicaciones SIG libres existentes.

Para ello se consideró la característica más destacable del software libre para SIG: su modularidad [52]; lo que favorece las interrelaciones y la reutilización de funcionalidades entre proyectos. Además, en el análisis se tuvo en cuenta el proceso de migración hacia software libre en el que se encuentran inmersas las empresas cubanas.

Software GRASS

Geographic Resources Analysis Support System (**GRASS**, por sus siglas en inglés) es el proyecto SIG libre más antiguo, con un desarrollo de más de 20 años. Su principal característica es su gran número de funcionalidades y su estructura modular favorece que los desarrolladores aporten al proyecto contribuciones individuales centradas en un campo concreto de aplicación. El mayor problema que presenta es su complejidad y su curva de aprendizaje; aun siendo un software muy potente, carece de una interfaz amigable y no está diseñado para ser empleado en un entorno de producción. La aparición de herramientas adicionales que facilitan el acceso a la potencia de GRASS, especialmente en el campo del análisis, está cambiando esta situación. Dentro de estas herramientas, *Quantum GIS* (**Qgis**) es la más destacable, constituyendo una interfaz de usuario sencilla para GRASS [53].

Software SAGA

System for Automated Geoscientific Analyses (**SAGA**, por sus siglas en inglés) es un software SIG de escritorio, multiplataforma, desarrollado en Alemania y con un fuerte enfoque hacia el análisis de datos geo-espaciales. SAGA incluye diversos algoritmos y una interfaz de desarrollo que facilita la programación de nuevas funcionalidades de análisis, siendo esta la mayor potencialidad del programa. Otras capacidades, tales como la creación de cartografía o la edición, se encuentran presentes pero muy poco desarrolladas y con escasa funcionalidad, evidenciando que el principal objetivo de este software es servir como herramienta de análisis [54].

Software Qgis

Qgis es una aplicación SIG con grandes potencialidades para la edición de mapas, multiplataforma y desarrollado utilizando Qt Toolkit[4] y C++. Ofrece muchas características SIG, entre las que se encuentran [55]:

- Permite crear, editar, administrar y exportar mapas vectoriales en varios formatos.
- Permite realizar análisis de datos espaciales de PostgreSQL/PostGIS usando el complemento de Python fTools.

[4] Disponible en: "https://es.opensuse.org/Qt_Toolkit"

- Incorpora a través de las herramientas de procesado, decenas de comandos de GRASS y SAGA para realizar análisis espacial tanto con datos vectoriales como ráster.
- Permite la integración de componentes desarrollados en Python a través del módulo PyQgis.
- Presenta una interfaz amigable.

A partir de las características que presentan las herramientas SIG analizadas, se concluye que Qgis presenta diversas funcionalidades que pueden ser utilizadas en el desarrollo de la solución. Se tuvo en cuenta principalmente su capacidad en cuanto al manejo de la cartografía, por lo que facilita la representación en mapas temáticos como resultado de los procesos asociados a la georreferenciación y análisis.

Además se consideró que presenta una interfaz amigable, permitiendo agilizar el proceso de aprendizaje de la herramienta. Por último se identificó que cuenta con un módulo para la integración de complementos, permitiendo la reutilización de algunas de sus funcionalidades e integración de la solución.

1.3. Herramientas, lenguajes y tecnologías a utilizar

En todo proceso investigativo es necesario la utilización de sistemas que permitan dar soporte, organizar, facilitar, agilizar y automatizar las tareas generadas durante el transcurso de la investigación. Las herramientas, lenguajes y tecnologías empleadas que se describen a continuación son de vital importancia para su correcta realización.

1.3.1. Lenguaje de modelado

UML es el acrónimo de Lenguaje Unificado de Modelado, este es el lenguaje estándar para visualizar, especificar, construir y documentar los artefactos de un sistema, utilizándose para el modelado del negocio y sistemas de software [56]. También ofrece un estándar para describir los modelos, incluyendo aspectos conceptuales como procesos de negocio, funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

1.3.2. Herramienta *Computer Aided Software Engineering* (CASE)

Las herramientas CASE son 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 para el desarrollo de un software [57].

Visual Paradigm

Es una herramienta de diseño que facilita el desarrollo de software. Ofrece un paquete completo útil para la captura de requisitos, la planificación del software, la planificación de pruebas, el modelado de clases y el modelado de datos [58].
Las principales características de la herramienta son:

- Soporta las últimas versiones del UML.
- Posee un poderoso generador de documentación y reportes en formato PDF, HTTP y JPG.
- Proporciona soporte para varios lenguajes en la generación de código e ingeniería inversa como: Java, C++, CORBA IDL, PHP, Ada y Python.
- Disponibilidad en múltiples plataformas (Windows, Linux)
- Capacidades de ingeniería directa e inversa.

Se selecciona Visual Paradigm for UML en su versión 8.1 como herramienta para el modelado UML, pues permite trabajar de forma colaborativa, hacer un trabajo organizado y ágil. Posibilita la realización de los diagramas necesarios para el desarrollo y mejor entendimiento de la aplicación. Permite realizar ingeniería inversa a partir del código fuente. Al ser seleccionado el lenguaje de modelado UML, es conveniente tener en cuenta su vinculación con Visual Paradigm, resaltando que este último presenta abundante documentación y demostraciones interactivas.

1.3.3. Lenguaje de programación

Los lenguajes de programación son un conjunto de símbolos junto a un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Constan de un léxico, una sintaxis y una semántica [59]. Partiendo de las características de la aplicación, se hace necesaria la selección de un lenguaje mediante el cual

se pueda cumplir con los requisitos propuestos. Actualmente existen muchos lenguajes para el desarrollo de aplicaciones, surgidos a partir de las tendencias y necesidades de los escenarios. El análisis se centró fundamentalmente en el lenguaje Python a partir de la posibilidad que brinda QGis para integrar componentes implementados en este lenguaje.

Python

Se trata de un lenguaje interpretado o de *script*, con tipado dinámico, multiplataforma y orientado a objetos, que permite la programación imperativa, funcional y orientada a aspectos [60].

Se seleccionó Python en su versión 2.7.5 porque es un requerimiento de QGis, su sintaxis es simple, clara y sencilla logrando de esta manera que los programas elaborados en este lenguaje parezcan pseudocódigo. Además el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, entre otros, hacen que desarrollar una aplicación en Python sea sencillo y rápido.

Es importante tener en cuenta que al seleccionar Qgis como el software que soportará la integración de la solución, el lenguaje de programación más eficiente y conveniente para utilizar es Python; este SIG a partir de su versión 0.9 trae soporte del lenguaje Python que junto con el módulo PyQT4 entrega una solución óptima al desarrollo de plugins e interfaces gráficas de usuario.

Scikit-learn

Scikit-learn es una biblioteca de aprendizaje automático, desarrollada sobre biblioteca matemáticas en Python, quizás la mejor opción si se desea hacer clasificación con este lenguaje. En particular se enfoca en los denominados clasificadores: máquinas de soporte vectorial y bosques aleatorios. Integra una amplia gama de algoritmos de aprendizaje de máquina, para problemas semi-supervisados y no supervisados. Este paquete se centra en ofrecer aprendizaje de máquina a los no especialistas incorporando el uso de un lenguaje de alto nivel con propósito general. Se hace énfasis en la facilidad de uso, rendimiento, documentación, y la coherencia de la interfaz de programación de aplicaciones (API del inglés: *Application Programming*

Interface). Cuenta con dependencias mínimas y se distribuye bajo la licencia *Berkeley Software Distribution* (BSD en por su siglas en ingles, distribución de software Berkeley) simplificada, fomentando su uso tanto en entornos académicos y comerciales. El código fuente, los binarios y la documentación se puede descargar de: [61].

PyQt

PyQt es un conjunto de enlaces Python para la biblioteca gráfica Qt. El módulo está desarrollado por la firma británica Riverbank Computing y se encuentra disponible para Windows, GNU/Linux y Mac OS bajo diferentes licencias. PyQt se distingue por su sencillez, por poseer un número importantes de herramientas que gestionen su manipulación y por su posibilidad de adecuarse a las distintas plataformas de software.
Utilizando PyQt en su versión 4.0 en el desarrollo de la herramienta informática, se puede crear una interfaz visual sencilla y sin muchos contratiempos, ya que PyQt posee los componentes visuales necesarios para su desarrollo, así como una abundante documentación y ejemplos.

Qt Designer

Qt Designer es una herramienta que permite acelerar el desarrollo de interfaces multilenguaje debido a que genera un archivo XML cuyo contenido es el formato de dicha interfaz, pudiéndolo convertir con los programas pertinentes a cada lenguaje. Esta herramienta provee características muy poderosas como la previa visualización de la interfaz, soporte para widgets y un editor de propiedades con gran variedad de opciones.
En correspondencia con la elección anterior de PyQt, se ha decidido emplear Qt Designer en su versión 4.7.4 como elemento que soportará el diseño de las interfaces. Su utilización permite la creación de las interfaces visuales de la aplicación de forma sencilla, además de la fácil manipulación de las variables de configuración de cada una de ellas.

1.3.4. Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE, por sus siglas en inglés) es una herramienta que permite a los desarrolladores de software escribir sus programas en uno o más lenguajes.

Consiste básicamente en una plataforma en la que se integran un editor de código, un compilador, un depurador y una interfaz gráfica de usuario (Entornos de programación 2012).

Pycharm

Pycharm es un editor de código inteligente que proporciona soporte de primera clase para los lenguajes de programación: Python, JavaScript, CoffeeScript, TypeScript, HTML/CSS, Cython, lenguajes de plantilla, AngularJS y Node.js, y otros menos utilizados. Pycharm funciona en las plataformas Windows, Mac OS y Linux con una única clave de licencia, también ofrece un espacio de trabajo con colores personalizables y atajos de teclado.

La decisión de seleccionar como IDE, Pycharm en su versión 3.4, está dada a que ofrece auto-completación inteligente de código, comprobación de errores sobre la marcha, soluciones rápidas y fácil navegación en el proyecto. Pycharm mantiene la calidad del código bajo control con chequeos, asistencia a pruebas, refactorizaciones inteligentes, y una serie de inspecciones, lo que ayuda a escribir un código limpio y fácil de mantener [62].

1.3.5. Gestor de base de datos

Los Gestores de Bases de Datos (**GBD**) permiten crear y mantener una base de datos, además actúan como interfaz entre los programas de aplicación y el sistema operativo. El objetivo principal es proporcionar un entorno eficiente a la hora de almacenar y recuperar la información de las base de datos. Estos softwares facilitan el proceso de definir, construir y manipular bases de datos para diversas aplicaciones [63].

PostgreSQL

PostgreSQL es un sistema de GBD objeto-relacional, de propósito general, multiusuario y de código abierto, que soporta gran parte del estándar SQL y ofrece modernas características como consultas complejas, disparadores, vistas, integridad transaccional, control de concurrencia multiversión. Puede ser extendido por el usuario añadiendo tipos de datos, operadores, funciones agregadas, funciones ventanas y funciones recursivas, métodos de indexado y lenguajes procedurales [64].

Fue seleccionado PostgreSQL en su versión 9.0, teniendo en cuenta que es un GBD multiplataforma y de código abierto. Además se valoró la existencia de la extensión PostGIS para permitir el trabajo con datos espaciales.

PostGIS

Para añadir soporte a PostgreSQL de objetos geográficos se utilizó la herramienta PostGIS en su versión 2.1.5. Este módulo convierte la base de datos objeto-relacional PostgreSQL en una base de datos espacial para su utilización en SIG.

PostGIS incluye un conjunto de operaciones para realizar consultas espaciales muy bien optimizadas por sus índices R-Tree y su integración con el planificador de consultas de PostgreSQL. Utiliza las librerías Proj4 para dar soporte a la transformación dinámica de coordenadas y la biblioteca GEOS para realizar operaciones de geometría. Utiliza bloqueo a nivel de fila, permitiendo a múltiples procesos trabajar con las tablas espaciales concurrentemente y asegurando la integridad de los datos [65].

PgAdmin

Como aplicación gráfica para gestionar el GBD PostgreSQL se utilizó la herramienta PgAdmin III en su versión 1.20.0. PgAdmin está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. Soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código para la parte del servidor y un agente para lanzar scripts programados. La conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas Unix), y puede encriptarse mediante SSL para mayor seguridad [66].

1.4. Metodología de desarrollo

El desarrollo de un software no es una tarea fácil, se debe contar con un proceso bien detallado, para esto se necesita aplicar una metodología que sea capaz de llevar a cabo el control total del producto. 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 de software. Dichas metodologías pretenden guiar a los desarrolladores, sin embargo los requisitos de un software son muy variados y cambiantes, y se ha dado lugar a que exista una gran variedad de ellas [67].

Las metodologías se dividen en dos grupos, tradicionales (pesadas) y ágiles (ligeras). Las tradicionales, se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, pretendiendo prever todo de antemano, además dependen de un equipo de desarrollo bastante grande. En las ágiles es más importante lograr que un producto de software se desarrolle con la calidad requerida, que realizar una buena documentación. En este tipo de metodología el cliente está presente en todo momento y colabora con el proyecto, que posee un equipo de desarrollo pequeño [67].

1.4.1. Programación Extrema

Programación extrema (**XP**, por sus siglas en inglés) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. Además se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico [68].

Características de la metodología XP [69]:

- XP es una metodología “liviana” que no tiene en cuenta la utilización de casos de uso y la generación de una extensa documentación.
- XP tiene asociado un ciclo de vida y es considerado a su vez un proceso.
- La tendencia de entregar software en espacios de tiempo cada vez más pequeños con exigencias de costos reducidos y altos estándares de calidad.
- XP define Historias de Usuario (**HU**) como base del software a desarrollar, estas historias las escribe el cliente y describen escenarios sobre el funcionamiento del programa. A partir

de las HU y de la arquitectura perseguida se crea un plan de liberaciones entre el equipo de desarrollo y el cliente.

Fases de la metodología XP:

- Planificación: durante esta etapa se lleva a cabo el proceso de identificación y confección de las HU.
- Diseño: durante esta etapa se crea un diseño evolutivo que va mejorando incrementalmente y que permite hacer entregas pequeñas y frecuentes de valor para el cliente, basado principalmente en el desarrollo de las tarjetas Clase-Responsabilidad- Colaboración (**CRC**).
- Desarrollo: en esta fase se realiza la implementación de las HU que fueron seleccionadas por cada iteración. Al inicio se lleva a cabo un chequeo del plan de iteraciones por si es necesario realizar modificaciones. Como parte de este plan se crean tareas de ingeniería para ayudar a organizar la implementación exitosa de las HU.
- Pruebas: esta fase permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñadas por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñada por el cliente final.

El ciclo de desarrollo consiste en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.

2. El programador estima el esfuerzo necesario para su implementación.

3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.

4. El programador construye ese valor de negocio.

5. Vuelve al paso 1

A partir del estudio de XP, se concluye que responde a las necesidades principales de tiempo, entorno y cantidad de programadores, e incluye al cliente como parte fundamental del equipo de desarrollo. Además, se preocupa más en el avance exitoso del producto que en generar una

documentación detallada del mismo, siendo capaz de adaptarse a los cambios de requisitos en cualquier punto del ciclo de vida del proyecto. Estos elementos demuestran que es una metodología factible para guiar el proceso de desarrollo de la solución, por lo que se decide incluir en la propuesta.

1.5. Conclusiones Parciales

Con el desarrollo de este capítulo se obtuvo un mejor dimensionamiento del problema a partir del análisis de los principales conceptos asociados a su solución. El análisis de los elementos básicos así como algunas técnicas de la Inteligencia Artificial, permitió conocer las características que poseen para darle solución al problema planteado. La revisión del panorama actual de los SIG condujo a la selección de QGis como el más adecuado para la integración de la propuesta. A partir del análisis de las herramientas y tecnologías se seleccionaron un conjuntos de ellas, basadas en licencia libre, para obtener un producto de alta independencia tecnológica y multiplataforma. La metodología XP se escogió para guiar el proceso de desarrollo de la solución.

Capítulo 2

Sistema informático para la georreferenciación y análisis de la distribución espacial de tumores malignos

2. SISTEMA INFORMÁTICO PARA LA GEORREFERENCIACIÓN Y ANÁLISIS DE LA DISTRIBUCIÓN ESPACIAL DE TUMORES MALIGNOS

En este capítulo se describen los elementos relacionados con el desarrollo de un sistema informático para la georreferenciación y análisis de los tumores malignos. Se especifican los requisitos de software. Se obtienen los artefactos correspondientes a las fases de planificación y diseño de la metodología seleccionada. Se define la arquitectura y los principales patrones de diseño utilizados en el desarrollo de la solución. Se detallan las tareas de ingenierías que conforman cada HU definida en la fase de planificación. Se establece el estándar de codificación que se utilizará en el desarrollo de la solución.

2.1. Propuesta de georreferenciación y análisis

En la presente investigación los procesos de georreferenciación y análisis se desglosa en las fases siguientes:

- Registro: se realiza la georreferenciación de los casos.
- Selección: se eligen un algoritmo para realizar el agrupamiento.
- Selección de parámetros: se realiza la selección de los parámetros según el algoritmo de agrupamiento.
- Agrupamiento: se clasifican los casos en diferentes conglomerados.
- Visualización: se representa en un mapa temático cada uno de los conglomerados.

En la figura 2.1 se presenta el modelo conceptual que guiará la propuesta de solución. A partir de los elementos del modelo y las características tanto de los tumores malignos, como de las técnicas de análisis espacial estudiadas en la literatura se identificaron los requisitos que se presentan en el epígrafe siguiente.

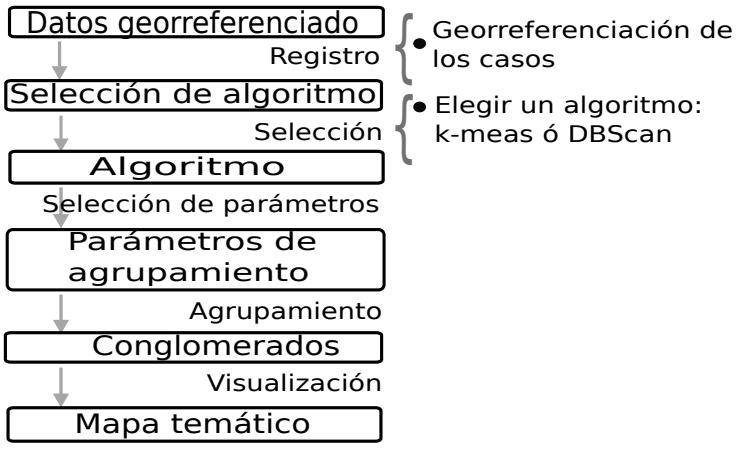

Figura 2.1: *Modelo conceptual de la propuesta de solución.*

2.2. Requisitos de software

"Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste" [70]. La calidad con que se realiza la captura de los requisitos incide en todo el proceso de desarrollo del software repercutiendo en el resto de las fases de su desarrollo. Además contribuye a tomar mejores decisiones de diseño y arquitectura.

2.2.1. Requisitos funcionales

Un requisito funcional (**RF**) define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requerimientos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone que un sistema debe cumplir. Estos son complementados por los requisitos no funcionales, que se enfocan en cambio, en el diseño o la implementación [70].

A continuación se muestran los RF identificados:

- **RF 1:** Georreferenciación de caso.
  - **RF 1.1:** Adicionar caso.
  - **RF 1.2:** Buscar caso.

- **RF 2:** Georreferenciar de caso aleatorio
- **RF 3:** Exportar registro de casos (xml, xls).
- **RF 4:** Importar registro de casos (xml, xls).
- **RF 5:** Estratificar mortalidad.
- **RF 6:** Detectar conglomerado.
- **RF 7:** Construir diagrama Hub para el análisis de vecinos más cercano.

2.2.2. Requisitos no funcionales

Los requisitos no funcionales (**RNF**) son propiedades o cualidades que el sistema debe tener. Estas propiedades o cualidades se refieren a las características que hacen al sistema estable, usable, rápido, confiable y escalable [70]. A continuación se muestran los RNF identificados:

Requisitos de Software

- **RNF 1:** Se debe tener instalada la herramienta QGIS en su versión 2.6 o superior.
- **RNF 2:** Se debe tener instalado el GBD PostgreSQL en su versión 9.0 o superior.
- **RNF 3:** Se debe tener instalado el módulo Postgis en su versión 2.1.5 o superior.

Requisitos de Hardware

- **RNF 4:** La estación de trabajo debe contar con al menos 1,0 GB de *Random Access Memory* (**RAM**, por sus siglas en inglés).
- **RNF 5:** La capacidad mínima de espacio en disco debe ser 2.0 GB.

Requisitos de Usabilidad

- **RNF 6:** Debe tener una interfaz gráfica, visualmente atractiva para el usuario. La aplicación podrá ser usada por cualquier usuario con conocimientos básicos sobre geografía e informática. Debe mostrar mensajes al usuario que le ayuden a llevar a cabo la tarea que realiza.

Requisitos de Interfaz

- **RNF 7:** Debe tener una interfaz amigable y con apariencia profesional.

- **RNF 8:** La interfaz debe tener un diseño sencillo y ser de fácil comprensión para el usuario.

Restricciones de diseño e implementación

- **RNF 9:** Se hace uso de la herramienta Qgis en su versión 2.6 e IDE Pycharm versión 3.4.
- **RNF 10:** El lenguaje de programación usado para la implementación es Python.

2.3. Fase de planificación

La metodología XP define como fase inicial la planificación. Durante esta etapa se lleva a cabo el proceso de identificación y confección de las historias de usuario, así como la familiarización del equipo de trabajo con las tecnologías y herramientas seleccionadas para el desarrollo del software. El cliente especifica la prioridad en que se deben implementar las historias de usuario, además de una estimación del esfuerzo. El resultado de la fase es un plan de entregas donde se realiza una estimación de las versiones que tendrá el producto en su realización, de manera tal que guíe su desarrollo [69].

2.3.1. Historias de usuarios

Las HU constituyen la técnica utilizada en XP para especificar los requisitos del software; en ellas el cliente describe brevemente las características que el sistema debe poseer, y se realiza una por cada característica principal del sistema. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento pueden reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas [67].
Luego de obtener las principales funcionalidades del sistema, se identificaron 8(ocho) HU. En las tablas 2.1 y 2.2 se muestra una breve descripción de 2(dos) de ellas.

Historia de Usuario "Adicionar caso"	
Número: 1.1	Nombre Historia de Usuario: Adicionar nuevo caso
Usuario: Experto	
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Puntos Estimados: 2	Iteración Asignada: 1
Programador responsable: Raniel Gé Vaillant	
Descripción: La aplicación debe ser capaz de adicionar nuevo caso, a partir de: 1-) Los datos de cada nuevo caso a insertar en el sistema. 2-) Todos los los campos son obligatorios.	
Observaciones:	

Tabla 2.1: *Historia de usuario: Adicionar caso.*

Historia de Usuario " Detectar conglomerado"	
Número: 4	Nombre Historia de Usuario: Detectar conglomerado
Usuario: Experto	
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Puntos Estimados: 2	Iteración Asignada: 2
Programador responsable: Raniel Gé Vaillant	
Descripción: La aplicación debe ser capaz de analizar los datos y permitir detectar conglomerado, a partir de: 1-) El análisis de los casos georreferenciados. 2-) A través de los algoritmos de detección de conglomerados a utilizar.	
Observaciones:	

Tabla 2.2: *Historia de usuario: Detectar conglomerado.*

2.3.2. Estimación de esfuerzos por historias de usuario

En el presente epígrafe se realiza la estimación del esfuerzo por HU, se hace necesario tener en cuenta que estas deben ser programadas en un tiempo de una a tres semanas. Si la estimación es superior a tres semanas, se divide en dos o más HU. Si es menor de una semana, se combina con otra HU. Estas estimaciones permiten tener una medida de la velocidad del proyecto y ofrecen una guía a la cual ajustarse. Los resultados estimados se muestran en la tabla 2.3.

Historia de usuario	Puntos de Estimación (semanas)
HU 1.1: Adicionar caso	2
HU 1.2: Buscar caso	1
HU 2: Georreferenciar caso aleatorio	2
HU 3: Exportar registro de caso (sexmo, vals).	1
HU 4: Importar registro de caso (sexmo, vals).	1
HU 5: Estratificar mortalidad.	2
HU 6: Detectar conglomerado.	2
HU 7: Construir diagrama Hub para analizar Los vecinos más cercanos	1

Tabla 2.3: *Estimación de esfuerzos por Historia de Usuario.*

2.3.3. Plan de iteraciones

Una vez finalizadas las HU se debe crear un plan de iteraciones, indicando cuáles se desarrollarán en cada iteración. En la tabla 2.4 se muestra cómo quedó definido el plan de iteraciones para la solución propuesta.

Iteraciones	Orden de las historias de usuario a Implementar	Duración de las iteraciones (sem)
Iteración 1	Georreferenciación de caso	3
Iteración 2	Estratificar mortalidad	2
Iteración 3	Georreferenciación de caso aleatorio	2
Iteración 4	Construir diagrama Hub para analizar los vecinos más cercanos	1
Iteración 5	Detectar conglomerado.	2
Iteración 5	Exportar registro de caso (xml, xls).	2
	Importar registro de caso (xml, xls).	
Total		**12**

Tabla 2.4: *Plan de duración de las iteraciones.*

2.3.4. Plan de entrega

El plan de entregas establece qué HU serán agrupadas para conformar una entrega, y el orden de implementación [68]. En este plan se concentran las funcionalidades referentes a un mismo

tema en módulos, esto permite un mayor entendimiento en la fase de implementación. Tiene como objetivo definir el número de liberaciones que se realizarán en el transcurso del proyecto y las iteraciones que se requieren para desarrollar cada una. De esta forma se puede trazar el plan de entrega en función de estos dos parámetros: el tiempo de desarrollo ideal y el grado de importancia para el cliente. En la tabla 2.5 se presenta el plan de entregas de la aplicación informática propuesta.

	Final de la 1ra Iteración	**Final de la 2da Iteración**	**Final de la 3ra Iteración**	**Final de la 4ta Iteración**	**Final de la 5ta Iteración**	**Final de la 6ta Iteración**
Módulos	2da semana de marzo	4ta semana de marzo	2da semana de abril	3ra semana de abril	1ra semana de mayo	2da semana de mayo
Georrefe-renciación y análisis	v1.0	v1.1	v1.2	v1.3	v1.4	Finalizado

Tabla 2.5: *Plan de duración de las entregas.*

2.4. Fase de diseño

La metodología de desarrollo XP plantea prácticas especializadas que accionan directamente en la realización del diseño para lograr un sistema robusto y reutilizable. Se trata en todo momento de conservar su simplicidad, es decir, crear un diseño evolutivo que va mejorando incrementalmente y que permite hacer entregas pequeñas y frecuentes de valor para el cliente, basado principalmente en el desarrollo de las tarjetas CRC.

2.4.1. Tarjetas Clase-Responsabilidad-Colaboración

Las tarjetas CRC son utilizadas para representar las responsabilidades de las clases y sus interacciones. Estas tarjetas permiten trabajar con una metodología basada en objetos, permitiendo que el equipo de desarrollo completo contribuya en la tarea del diseño. En cada tarjeta CRC el nombre de la clase se coloca a modo de título, las responsabilidades se colocan a la izquierda y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente.

Una clase es cualquier persona, evento, concepto, pantalla o reporte. Las responsabilidades de

una clase son las cosas que conoce y las que realizan, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades [71].

En las tablas 2.6 y 2.7 se muestran las tarjetas CRC correspondientes a las clases.

Clase: NuevoCaso	
Responsabilidad	Colaboración
Crear instancias de la clase Caso.	GCasos, Caso

Tabla 2.6: *Tarjeta CRC para la clase NuevoCaso.*

Clase: Algoritmo	
Responsabilidad	Colaboración
Crear instancias de las clases Kmeas y DBScan.	Conglomerado, ControladorAnalisis

Tabla 2.7: *Tarjeta CRC para la clase Algoritmo.*

2.5. Arquitectura de software

La arquitectura de software es la definición y estructuración de una solución que cumple con los requisitos técnicos y operativos. Optimiza atributos que implican una serie de decisiones, tales como la seguridad, el rendimiento y la capacidad de administración. Estas decisiones en última instancia, afectan la calidad del producto, el mantenimiento, rendimiento y éxito global [72].

2.5.1. Estilo arquitectónico a utilizar

Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software. Estos permiten expresar un esquema de organización estructural esencial para un sistema de software [72].

En este trabajo se hace uso del estilo arquitectónico modelo-vista-controlador, logrando que el sistema quede organizado y así tener un orden lógico en la programación del mismo.

Modelo Vista Controlador:

El Modelo Vista Controlador(*Model View Controller*, **MVC** por sus siglas en inglés) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la interfaz de usuario y el código es el que provee de datos dinámicos a la página; el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista.
En esta investigación se presenta el sistema (Sistema para el análisis espacial en salud **XANGEO**) que está compuesto por los módulos:

- Común provee útiles para facilita la conexión con el SIG
- Los módulos Estratificador, Servicio y (Sistema para la georreferenciación y análisis de los tumores malignos **X-GATMa**) en los cuales se evidencia el patrón arquitectónico MVC como se muestra en la figura 2.2

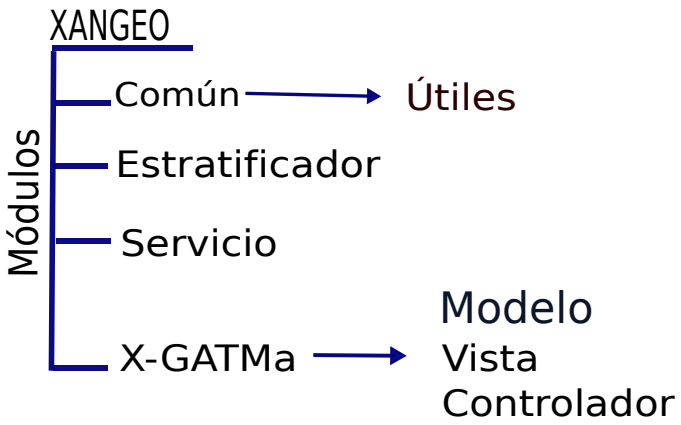

Figura 2.2: *Evidencia de la arquitectura del sistema.*

Modelo: Esta es la representación específica de la información con la cual el sistema opera. Maneja los datos y controla sus transformaciones. Este no tiene conocimiento específico de

los controladores y las vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas y notificar a las vistas cuando cambia el modelo.

Vista: Maneja la presentación visual de los datos representados por el modelo, el cual es presentado en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Responde a eventos, usualmente acciones del usuario, e invoca cambios en el modelo y probablemente en la vista.

2.6. Patrones de diseño

Los patrones de diseño constituyen la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo cual significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

2.6.1. Patrones de diseño GRASP

Los Patrones Generales de Software para la Asignación de Responsabilidades (**GRASP**, por sus siglas en inglés) son utilizados para describir los principios fundamentales del diseño y la asignación de responsabilidades [73]. Entre los que se utilizaron en la solución figuran los siguientes: Experto, Creador, Controlador, Bajo acoplamiento y Alta cohesión.

Experto: el patrón Experto define cómo asignar de forma adecuada las responsabilidades en un modelo de clases. Indica que la responsabilidad de la creación de un objeto o la implementación de un método debe recaer en la clase que conoce toda la información necesaria para crearlo. Dicho patrón se evidencia en la aplicación informática en la clase NuevoCaso, como esta posee toda la información necesaria para calcular el aporte de riesgo de cada territorio se le asigna dicha responsabilidad. En la figura 2.3 se muestra una imagen de dicha clase.

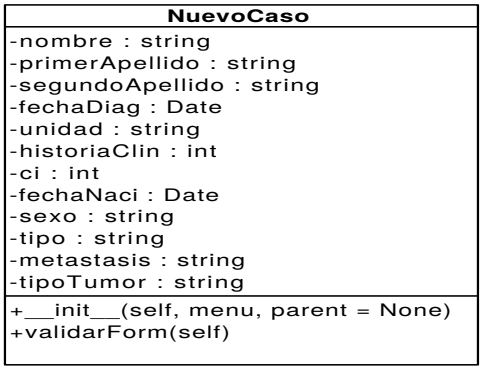

Figura 2.3: *Evidencia del patrón Experto.*

Creador: este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. Su intención es encontrar un creador que necesite conectarse al objeto creado en alguna situación. En la aplicación informática se pone de manifiesto dicho patrón en la clase GCasos, a esta se le asigna la responsabilidad de crear instancias de la clase NuevoCaso. En la figura 2.4 se presenta la evidencia de dicho patrón.

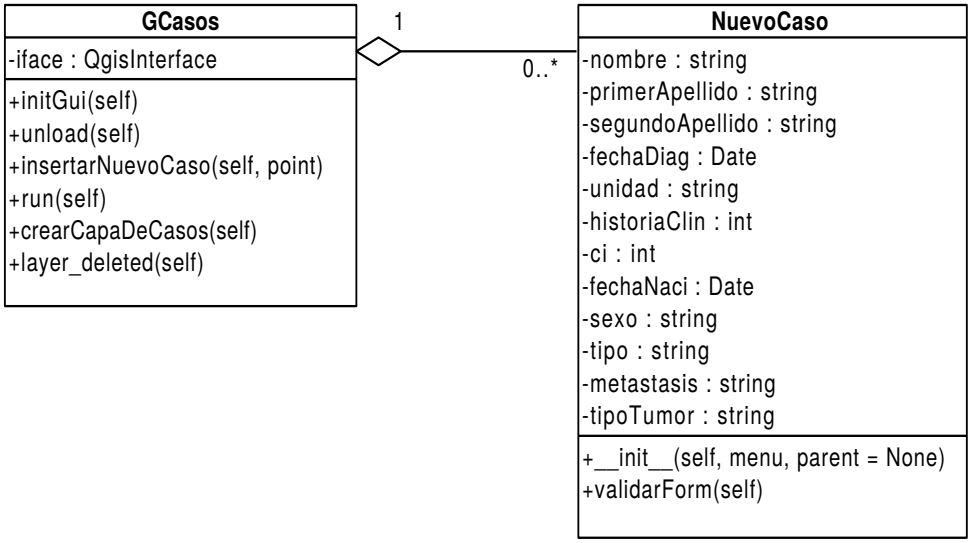

Figura 2.4: *Evidencia del patrón Creador.*

Controlador: permite asignar la responsabilidad de controlar el flujo de eventos del sistema1, a clases específicas, facilitando la centralización de actividades. El controlador no realiza estas

actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema. Este patrón se evidencia en la aplicación informática en la clase Controlador GCasos, a esta se le asignó la responsabilidad de manejar los eventos del sistema generados por el usuario. En la figura 2.5 se muestra una imagen de dicha clase.

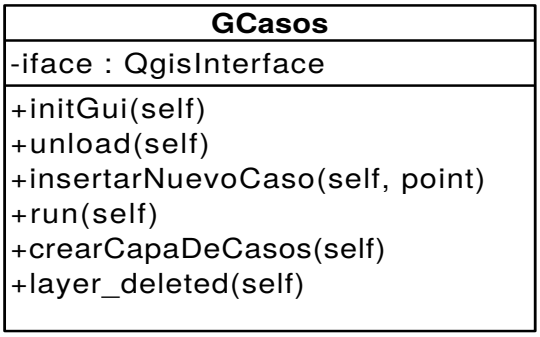

Figura 2.5: *Evidencia del patrón Controlador.*

Bajo acoplamiento: Este patrón se garantiza en la aplicación basándose en la propia arquitectura del sistema, lo que permite que las dependencias entre las clases sean muy pocas, ya que solamente las clases de una capa se pueden comunicar con las de la capa inmediatamente inferior.

Alta cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una baja cohesión hace muchas cosas no afines o realiza trabajo excesivo. En resumen, este patrón se observa cuando una clase tiene la responsabilidad de realizar una labor dentro del sistema, no desempeñada por el resto de los componentes del diseño. Este patrón se evidencia en la aplicación informática en conjunto con el patrón bajo acoplamiento, de forma tal que cada clase realice sus acciones y se evita que otra clase realice acciones correspondientes a la clase con la que está relacionada.

2.6.2. Patrones de GoF

Los Patrones del Grupo de Cuatro (**GoF**, por sus siglas en inglés) resuelven problemas específicos de diseño de software [74]. Estos patrones se agrupan en las siguientes categorías: creacionales, estructurales y de comportamiento.

- **Comportamiento:** Contribuyen a definir la comunicación e interacción entre los objetos de un sistema.
- **Creacionales:** Permiten abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
- **Estructurales:** Describen cómo las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades.

Método plantilla: es un patrón de comportamiento que define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos, esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar estructura. Este patrón se evidencia en las clases DBScan y KMeans, que heredan todas las funcionalidades de la clase Algoritmo, y redefinen los métodos distancia() y run() en función de sus características. En la figura 2.6 se muestra cómo se evidencia el patrón plantilla en la aplicación informática propuesta.

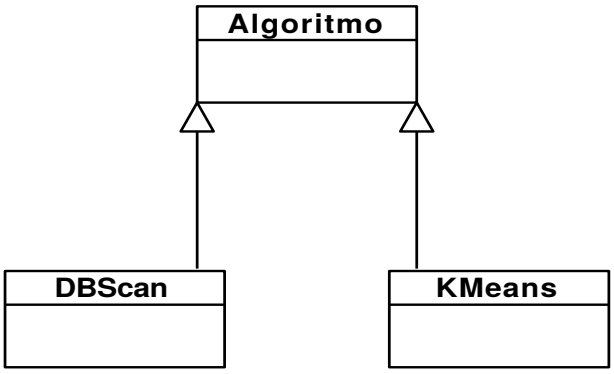

Figura 2.6: *Evidencia del patrón Plantilla.*

2.7. Diagrama de clases del diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación [73]. Un diagrama de este tipo presenta las clases del sistema con sus relaciones estructurales y de herencia. En la figura 2.7 se muestra el diagrama de clases de la aplicación informática propuesta.

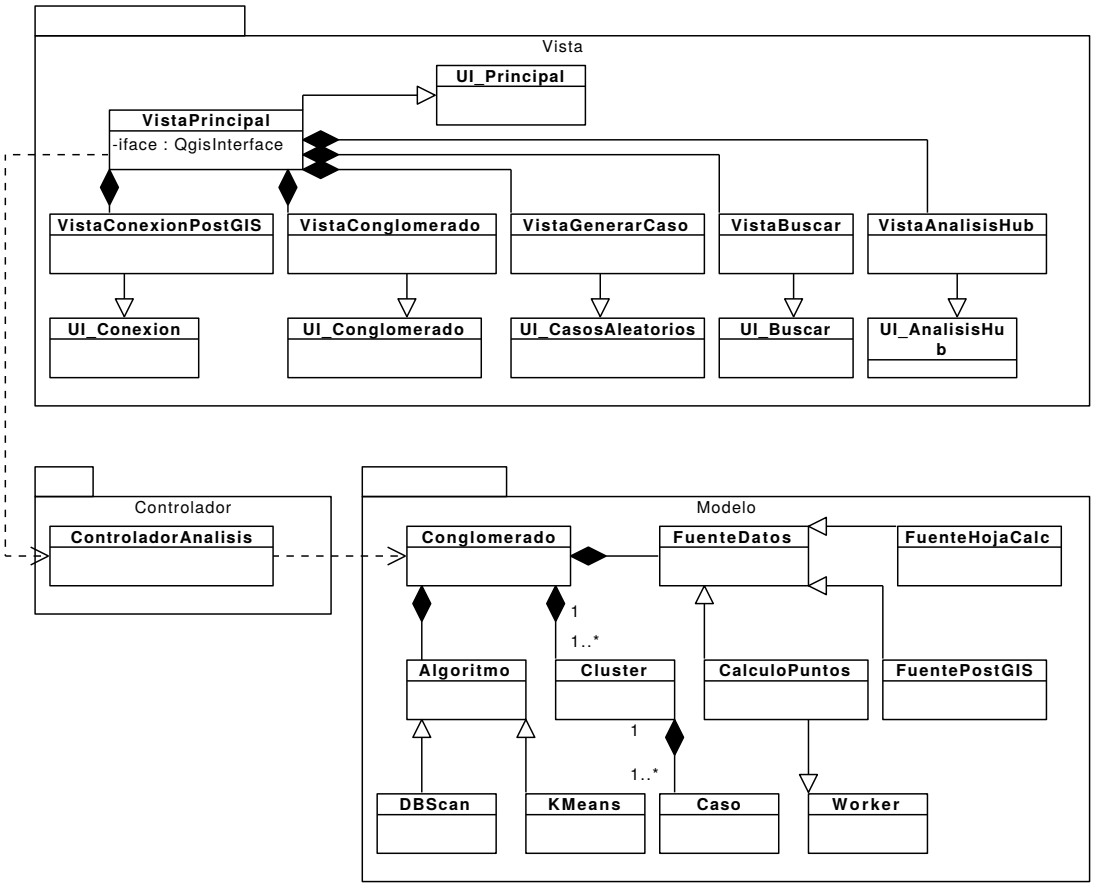

Figura 2.7: *Diagrama de clase.*

2.8. Fase de implementación

Luego de haber definido los elementos necesarios en la etapa de planificación y diseño se pasa a la de codificación o implementación de la aplicación, donde se da cumplimento al plan de iteraciones. En esta fase se realiza la implementación de las HU que fueron seleccionadas

por cada iteración, además se crean las tareas de ingeniería para ayudar a organizar la implementación exitosa de las HU. Al finalizar esta fase el cliente estará listo para comenzar a realizar las pruebas.

2.8.1. Tarea de ingeniería

Cada HU está compuesta por una o varias tareas de ingeniería, éstas se realizan para especificar las acciones llevadas a cabo por el programador responsable de ella o ellas. En la tabla 2.8 se detallan para la iteración número cinco, las tareas a desarrollar por cada HU y en la tabla 2.9 se describe una tarea de ingeniería que responde a una HU arquitectónicamente significativa.

HU	Tareas de ingeniería por HU
Detectar conglomerado	Agrupar los casos utilizando el algoritmo k-Meas. Agrupar los casos utilizando el algoritmo DBScan. Agrupar los casos utilizando el algoritmo propagación por afinidad. Obtener los conglomerados con el algoritmo k-Meas. Obtener los conglomerados con el algoritmo DBScan. Obtener los conglomerados con el algoritmo propagación por afinidad. Mostrar los conglomerados en el mapa temático.

Tabla 2.8: *Distribución de tareas de ingeniería por HU (iteración 5).*

Tarea de ingeniería	
Número Tarea: 2	Número Historia de Usuario: HU # 6
Nombre Tarea: Agrupar los casos utilizando el algoritmo DBScan.	
Tipo Tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 28/04/2015	Fecha Fin: 2/05/2015
Programador Responsable: Raniel Gé Vaillant	
Descripción: Esta tarea permite agrupar los casos georreferenciados por el usuario utilizando el algoritmo DBScan.	

Tabla 2.9: *Tarea de ingeniería Agrupar los casos utilizando el algoritmo DBScan.*

2.8.2. Estándares de codificación

XP resalta que la comunicación de los programadores es a través del código, por lo que es necesario que sigan ciertos estándares de programación para lograr un entendimiento entre los programadores, de manera que cualquier persona del equipo de desarrollo pueda modificar el

código. Además, se hace preciso que el código sea entendible para que posteriormente otros programadores puedan apoyarse en ese trabajo y desarrollen otras soluciones.
En el caso de la herramienta que se desarrolla, el estándar que se utiliza es:

Máxima longitud de las líneas

- Todas las líneas se limitan a un máximo de 79 caracteres.

Importaciones

- Las importaciones se encuentran en líneas separadas.

Comentarios

- Se utilizan comentarios de una línea para hacer más entendible el código.
  Comentarios de una línea: comentario pequeño que solo abarca una línea y describe el código que le sigue.

Estilo de los nombres

- **Clases e Interfaces:** los nombres de las clases presentan la primera letra en mayúscula, en caso de ser un nombre compuesto, la inicial de cada palabra se representa en mayúscula. Se utilizan nombres simples y de alguna manera que describan el contenido, se usan palabras completas, a no ser que la abreviatura sea muy conocida.
- **Métodos y variables:** los nombres de cada métodos se representan en minúscula, en caso de ser un nombre compuesto, la inicial de la primera palabra se simboliza en minúscula, y la de las otras palabras que lo componen en mayúscula. Los nombres de las variables son cortos pero con significados lógicos, capaces de permitir a un observador identificar su función.

2.9. Conclusiones del capítulo

La propuesta de solución definida facilitará la realización de la georreferenciación y análisis de la distribución de tumores malignos utilizando SIG. La identificación de los requisitos permitió un mayor entendimiento de las necesidades del cliente. Mediante la descripción de las HU divididas

por iteraciones y la planificación del esfuerzo dedicado al desarrollo en cada una de ellas, se logró una mejor organización del trabajo y el establecimiento de fechas para la culminación por cada una de la iteración. El estilo arquitectónico MVC y de los patrones de diseño GRASP y GoF permitió una mejor estructuración de la aplicación. Por ultimo en la implementación se detallaron las tareas de ingeniería correspondiente a cada HU lo que facilitó la organización del trabajo en una secuencia lógica de pasos. El estándar de codificación utilizado proporcionó un buen entendimiento y una mejor organización del código.

Capítulo 3

Validación de la solución propuesta

3. VALIDACIÓN DE LA PROPUESTA

En el siguiente capítulo se realizan las pruebas definidas por la metodología seleccionada, se aplicaron pruebas unitarias para verificar el código y pruebas de aceptación para comprobar si al final de cada iteración se consiguió la funcionalidad requerida. Se realiza un caso de estudio para mostrar las potencialidades de la solución propuesta en el análisis de la distribución espacial de tumores malignos.

3.1. Fase de pruebas

Cuando se desarrolla una solución informática se deben realizar una gran cantidad de pruebas para verificar que el código esté correcto. Estas pruebas normalmente son ejecutadas en varias ocasiones y se ven afectadas por los cambios que se introducen conforme se va construyendo la solución. XP divide las pruebas en dos grupos: pruebas de aceptación, o pruebas funcionales diseñadas por el cliente final, destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida y pruebas unitarias, encargadas de verificar el código, diseñadas por los programadores.

3.1.1. Pruebas de aceptación

Las pruebas de aceptación en XP son especificadas por el cliente, y se centran en las características y funcionalidades generales del sistema, que son visibles y revisables por parte del usuario. Estas pruebas se derivan de las HU que se han implementado como parte de la liberación del software [68].

Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una HU no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de

aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información [68].

Casos de prueba

En las tablas 3.1 y 3.2 se muestran los casos de prueba de aceptación aplicados a las HU Adicionar caso y Detectar conglomerado.

Caso de prueba de aceptación	
Código: HU1.1_P1	Historia de Usuario: 1.1
Nombre: Adicionar caso.	
Descripción: Prueba para validar la funcionalidad adicionar caso.	
Condiciones de ejecución: El usuario debe en el sistema XANGEO seleccionar X-GATMa y la opción Georreferenciar nuevo caso. El usuario debe nombrar el nuevo punto. El usuario debe seleccionar la opción Aceptar El usuario selecciona Aceptar en la ventana Selector de sistema de referencia de coordenada. El usuario puede seleccionar sobre el mapa el punto a georreferenciar y se levanta una ventana con un formulario de los datos del caso . El usuario debe de llenar todos los campos del formulario para recoger la información del caso. El usuario debe seleccionar la opción Aceptar.	
Resultados esperados: En caso que se cumplan las condiciones de ejecución, el sistema adiciona el nuevo caso georreferenciado. En caso contrario el sistema muestra un mensaje informando el motivo por el cual no se pudo georreferenciar satisfactoriamente.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 3.1: *Caso de prueba de aceptación Adicionar caso.*

Caso de prueba de aceptación	
Código: HU6_P2	Historia de Usuario: 6
Nombre: Detectar conglomerado.	
Descripción: Prueba para validar la funcionalidad detectar conglomerado.	
Condiciones de ejecución: El usuario debe en el sistema XANGEO seleccionar X-GATMa y la opción Agrupamiento. El usuario debe seleccionar la capa donde están los casos, elige el algoritmo que utilizara para realizar el agrupamiento y luego le pasa los parámetros del algoritmo. El usuario debe seleccionar laopcion Aceptar	
Resultados esperados: En caso que se cumplan las condiciones de ejecución, el sistema construye los conglomerados. En caso contrario el sistema muestra un mensaje informando el motivo por el cual no realizó el agrupamiento.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 3.2: *Caso de prueba de aceptación Detectar conglomerado.*

Análisis de los resultados

Para validar que el resultado obtenido por el sistema coincide con el resultado esperado

por el cliente se diseñaron un total de 8 casos de prueba de aceptación en conjunto cliente-desarrolladores. De este total, 4 arrojaron el resultado esperado mientras que 4 pruebas resultaron fallidas, las funcionalidades que respondían a estas pruebas fueron tratadas en la siguiente iteración y al volver a aplicar las pruebas de funcionalidad mostraron un resultado de 7 exitoso y 1 fallida. Se realizó el tratamiento correspondiente a esa funcionalidad en la última iteración y al aplicar las pruebas se obtuvieron un total 8 pruebas satisfactorias para 8 casos de prueba aplicados, como se muestra en la figura 3.1.

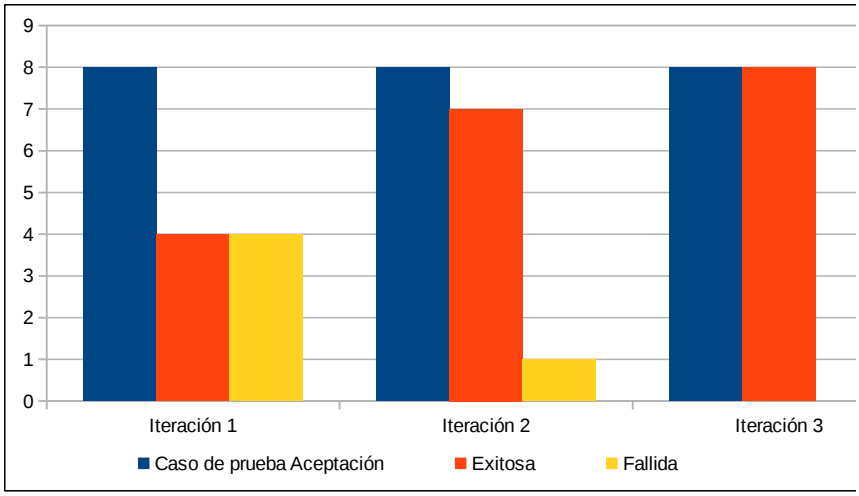

Figura 3.1: *Resultado de aplicar la prueba de aceptación.*

3.1.2. Pruebas de caja blanca

Las pruebas de caja blanca se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al *código fuente*. Se escogen distintos valores de entrada para examinar cada uno de los posibles *flujos de ejecución* del programa cerciorándose que se devuelvan los valores de salida adecuados [72]. Las pruebas de caja blanca intentan garantizar que:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecuten todos los bucles en sus límites.
- Se utilizan todas las estructuras de datos internas.

La técnica utilizada dentro de las pruebas de caja blanca fue camino básico. En la figura 3.2 se muestran las sentencias de código del método clusterCapa().

```
def clusterCapa(self,labels):
    self.capa.startEditing()
    d = self.capa.dataProvider()
    if self.capa.fieldNameIndex("Cluster") is -1:
        d.addAttributes([QgsField("Cluster", QVariant.Int)])
        self.capa.updateFields()
    p = 0
    m = self.capa.fieldNameIndex("Cluster")
    id=self.capa.fieldNameIndex("id")
    self.capa.startEditing()
    for i in range(len(labels)):
        self.capa.changeAttributeValue(i, m, int(labels[i]))
        self.capa.changeAttributeValue(i, id, i)
    self.capa.commitChanges()
```

Figura 3.2: *Código del método clusterCapa().*

Luego de haberse construido el grafo se realiza el cálculo de la complejidad ciclomática (es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa) mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad sea correcto.

1. La complejidad ciclomática coincide con el número de regiones del grafo de flujo.
2. La complejidad ciclomática, V(G), de un grafo de flujo G, se define como V(G) = Aristas - Nodos + 2.
3. La complejidad ciclomática, V(G), de un grafo de flujo G, también se define como V(G) = Nodos de predicado (es él que representa una condicional if o case, es decir, de él salen varios caminos.) 2 + 1.

A partir del grafo de flujo del método que se presenta en la figura 3.3, la complejidad ciclomática sería:

- Como el grafo tiene tres regiones, $V(G) = 3$

- Como el grafo tiene 8 aristas y 7 nodos, $V(G) = 8 - 7 + 2 = 3$
- Como el grafo tiene 2 nodos de predicado, $V(G) = 2 + 1 = 3$

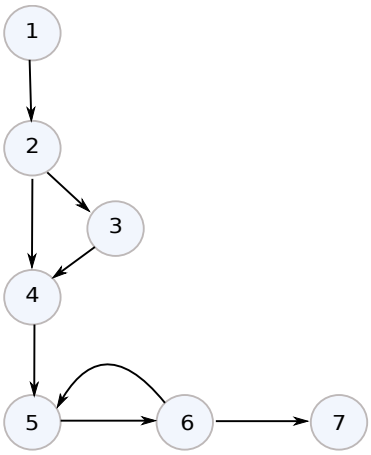

Figura 3.3: *Grafo de flujo del método*

Dado a que el cálculo de las tres fórmulas anteriormente mencionadas arrojó el mismo resultado se puede plantear que la complejidad ciclomática del método es 3. Esto significa que existen 3 posibles caminos por donde el flujo puede circular. Este valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

Caminos básicos identificados:

- Camino 1: 1-2-4-5-6-7
- Camino 2: 1-2-3-4-5-6-7
- Camino 3: 1-2-3-4-5-6-5-6-7

Para cada camino básico determinado se realiza un diseño de caso de prueba, como se muestra en las tablas 3.3, 3.4 y 3.5, que representan los caminos básicos 1, 2 y 3 respectivamente.

Caso de prueba para el camino básico #1 (1-2-4-5-6-7)	
Descripción	Prueba para validar los resultado del método clusterCapa(), se tiene un objeto y la capa con un atributo Cluster.
Condición de ejecución	Labels tiene 1 objeto y en capa se crea el atributo Cluster
Entrada	Labels y capa de punto
Resultado	Asociar al objeto geográfico el numero de cluster que le corresponde
Resultado de la prueba	Prueba satisfactoria

Tabla 3.3: *Caso de prueba para el camino básico # 1.*

Caso de prueba para el camino básico #2 (1-2-3-4-5-6-7)	
Descripción	Prueba para validar los resultados del método clusterCapa(), se tiene un objeto y la capa no cuenta con un atributo cluster.
Condición de ejecución	Labels tiene 1 objeto y la capa contiene el atributo Cluster
Entrada	Labels y capa de punto
Resultado	Asociar al objeto geográfico el numero de cluster que le corresponde
Resultado de la prueba	Prueba satisfactoria

Tabla 3.4: *Caso de prueba para el camino básico # 2.*

Caso de prueba para el camino básico #3 (1-2-3-4-5-6-5-6-7)	
Descripción	Prueba para validar los resultados del método clusterCapa(), se tiene dos objeto y la capa no cuenta con un atributo cluster.
Condición de ejecución	Labels tiene 2 objeto y en capa se crea el atributo Cluster
Entrada	Labels y capa de punto
Resultado	Asociar al objeto geográfico el numero de cluster que le corresponde
Resultado de la prueba	Prueba satisfactoria

Tabla 3.5: *Caso de prueba para el camino básico # 3.*

Análisis de los resultados Para la validación del código generado en el desarrollo de la herramienta se seleccionaron los métodos más relevantes, a los cuales se le realizaron las pruebas para poder evaluar si el funcionamiento de cada método se comportó de la manera esperada. Se realizaron un total de 13 pruebas a las 6 funcionalidades seleccionadas como relevantes, de las cuales 8 resultaron exitosas en una primera iteración de pruebas y 5 fallidas. En una segunda

iteración fueron tratadas y solucionadas no conformidades detectadas en la primera, se obtuvo el 100 % de pruebas exitosas, comprobándose la estabilidad de la lógica aplicada en el código generado en el desarrollo de la herramienta informática.

Para tener un mejor entendimiento de lo anterior se puede observar la figura 3.4 que se muestra a continuación:

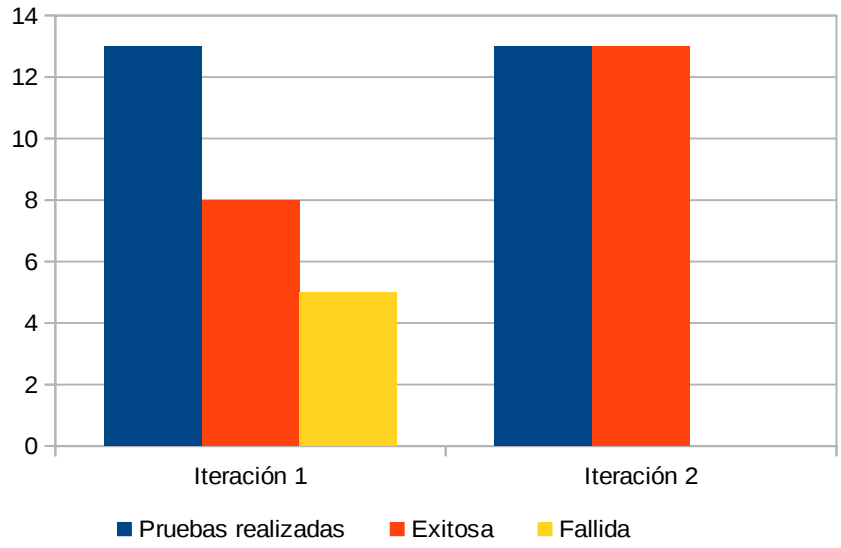

Figura 3.4: *Resultado de apllicar la prueba de caja blanca.*

3.1.3. Caso de estudio

Se realizó un estudio con el objetivo de verificar las potencialidades del uso de la propuesta como herramienta para el análisis. Para el conjunto de datos a emplear se creó una base artificial como se explica a continuación:

1. Se obtuvo del anuario estadístico del Ministerio de Salud Pública de Cuba para el año 2014 el total de casos registrados por provincia.
2. Desde los servicios que provee la Infraestructura de Datos Espaciales de la República de Cuba (IDERC)[75] se obtuvo la cartografía de las provincias de Cuba.
3. Utilizando el componente Generador de casos se ubicó el total de casos por provincias en la cartografía, ubicando en el polígono que representa a cada provincia una cantidad de

puntos igual a los casos reportados.

4. A partir de la densidad poblacional de la provincia La Habana y por los riesgos de salud identificados en otras investigaciones se decide utilizarla en el análisis [76].
5. Se realizó el agrupamiento de los casos utilizando el algoritmo *K Means* para ello, primeramente se estimó el valor de k y luego se construyeron los grupos.

Estimación del número de grupos

En los algoritmos de aprendizaje no supervisado, la cantidad de grupos puede ser un parámetro de entrada al algoritmo o puede ser determinado automáticamente por otros algoritmos. En el primer caso, como ocurre con el algoritmo K-Means, la estimación del número de *clusters* se obtiene a partir de alguna medida externa al algoritmo o con la ejecución de otro que sugiera la cantidad de grupos. En este estudio, primeramente se ejecutó el algoritmo propagación de afinidad que sugiere un número de grupos. Luego se construyeron los grupos utilizando K-Means y se comprobó a partir del coeficiente de silueta, los resultados se muestran más adelante.

El coeficiente de silueta es una métrica para evaluar la heterogeneidad de los grupos producidos por algoritmos de aprendizaje no supervisado. Un valor más alto de este índice indica un caso más deseable del número de *clusters*.

El coeficiente de Silueta para una observación i se denota como $s(i)$ y se define como:

$s(i) = \frac{b-a}{\text{máx}(a,b)}$

Donde: a es el promedio de las disimilitudes (o distancias) de la observación i con las demás observaciones del *cluster* al que pertenece i. Y b es la distancia mínima a otro *cluster* diferente al que contiene a la observación i. Ese *cluster* es la segunda mejor opción para i y se lo denomina vecindad de i.

El valor de $s(i)$ puede ser obtenido combinando los valores de a y b como se muestra a continuación:

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)} & si \quad a(i) < b(i) \\ 0 & si \quad a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & si \quad a(i) > b(i) \end{cases}$$

El coeficiente de Silueta es un valor comprendido entre -1 y 1, $-1 \leq s(i) \leq 1$.
Al analizar las posibles soluciones para que el coeficiente de Silueta sea cercano a 1, el valor de b tiene que ser mayor al de a. Esto significa que la distancia de la observación i a los *clusters* vecinos es suficientemente grande para que su pertenencia al *cluster* actual sea la correcta. Es decir, no es similar a sus vecinos. Un valor de $s(i)$ que sea cercano a cero nos va a indicar que la observación i está en la frontera de dos *clusters* y si el valor de s(i) es negativo, entonces la observación i debería ser asignada al *cluster* más cercano como se muestra a continuación:

- $s(i) \approx 1$, la observación i está bien asignada a su *cluster*.
- $s(i) \approx 0$, la observación i está entre dos *cluster*.
- $s(i) \approx -1$, la observación i está mal asignada a su *cluster*.

Podemos calcular el coeficiente de Silueta como el promedio de todos los $s(i)$ para todas las observaciones del conjunto de datos.
El algoritmo propagación de afinidad sugirió construir 63 grupos como se muestra en el mapa de la figura 3.5. Luego se realizó el agrupamiento utilizando K-Means para $k = 63$ y se calculó el coeficiente de silueta. Los resultados se muestran en la figuras 3.6 y 3.7.

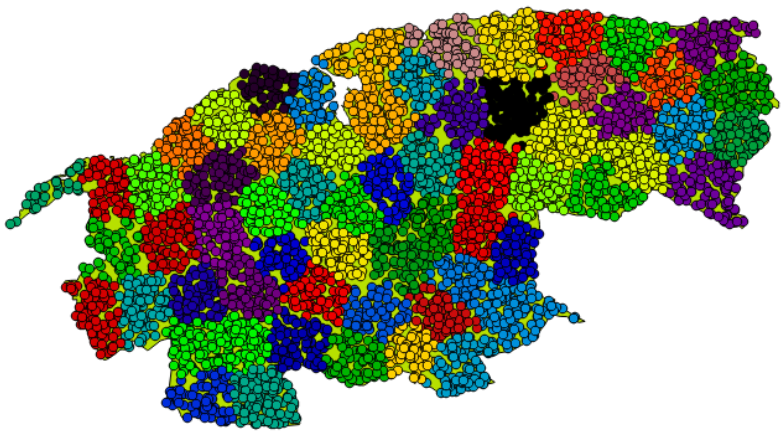

Figura 3.5: *Agrupamiento con propagación de afinidad*

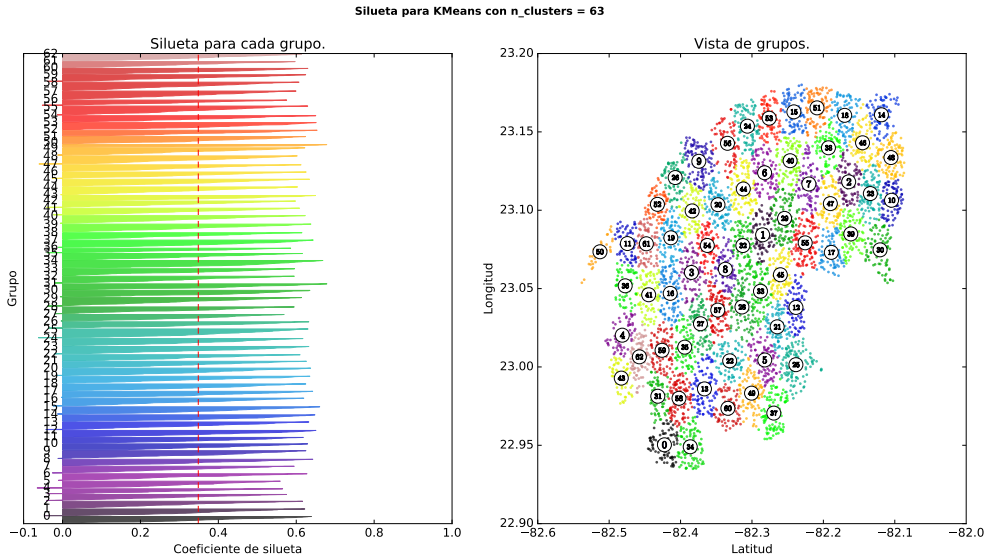

Figura 3.6: *Análisis de silueta para k=63*

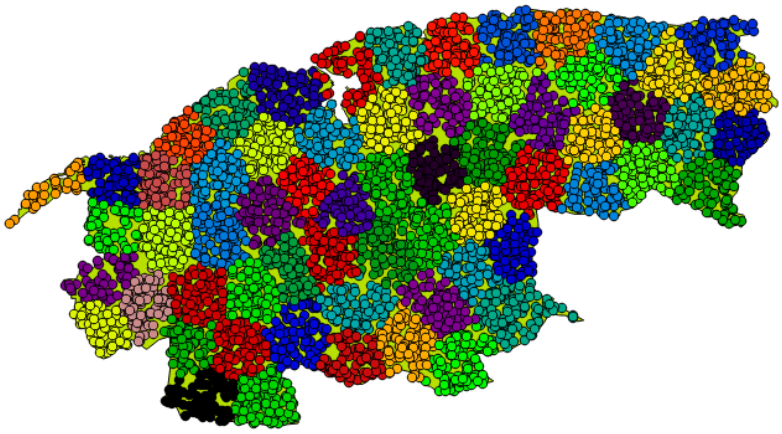

Figura 3.7: *Agrupamiento con k-means y k=63*

Finalmente se construyó un gráfico de distancias desde los municipios hasta los casos más cercanos como se muestra en la figura 3.8, que permita establecer relaciones entre municipios, posibles factores de riesgo y grupos obtenidos en el agrupamiento. Se puede visualizar que en algunos municipios los puntos más cercanos se encuentran en un grupo lo que sugiere la hipótesis de una concentración de factores de riesgos. En otros se dispersa en varios grupos pudiendo estar relacionado con varios factores de riesgos distribuidos en el municipio o cercanos

a él.

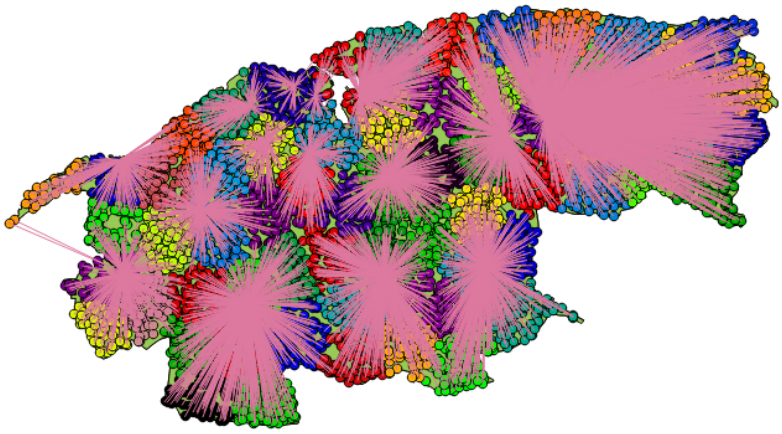

Figura 3.8: *Distancias mínimas entre municipios y casos*

3.2. Conclusiones del capítulo

En el presente capítulo con la aplicación de las pruebas de aceptación y caja blanca se pudo detectar, documentar y corregir las no conformidades existentes en el sistema implementado. La realización de estas pruebas permitió verificar el correcto funcionamiento del sistema y el cumplimiento de los requisitos del cliente. La realización del caso de estudio evidenció las potencialidades de la solución para el análisis espacial de la distribución de tumores malignos a partir de las técnicas de agrupamiento. Se mostró además la posibilidad de sugerir hipótesis relacionada con los factores de riesgos por municipios.

CONCLUSIONES

Como resultados de la presente investigación se obtuvo una propuesta de solución para la georreferenciación y análisis de los tumores malignos utilizando SIG que contribuye a la planificación de recursos para la prevención del cáncer. En función de los resultados obtenidos se arribó a las siguientes conclusiones:

1. La definición del marco teórico referencial de la investigación relacionado con la georreferenciación y análisis de los tumores malignos, fundamentaron la necesidad de desarrollar un plugin que se adapte a los objetivos expuestos y satisfaga las necesidades del país.
2. La integración de la solución propuesta al sistema QGis facilitó la realización de la georreferenciación y análisis de la distribución espacial de los tumores malignos para identificar regiones más expuestas a factores de riesgos.
3. Las pruebas aplicadas para la verificación de la solución informática y la valoración de los resultados a través de un caso de estudio demostraron que el sistema cumple con los requisitos definidos, garantizando su correcto funcionamiento.
4. La incorporación de técnicas de agrupamiento y el análisis de proximidad entre los casos y los municipios sugiere la conformación de hipótesis relacionada con la distribución de factores de riesgos por territorios.

RECOMENDACIONES

Al concluir esta investigación, se recomienda para futuros trabajos asociados a esta área del conocimiento:

1. Incorporar otros algoritmos de agrupamiento para realizar el análisis de los casos georreferenciados y comparar los resultados obtenidos por varios métodos.
2. Utilizar la herramienta con una base de datos real de casos y factores de riesgos que permita evaluar el riesgo al que se expone la población.
3. Incorporar otras técnicas de análisis espacial como el análisis de proximidad o la interpolación espacial que permitan contrastar resultados a partir de varios métodos.

REFERENCIAS BIBLIOGRÁFICAS

[1] Ramos, M. Distribución geográfica de algunos tumores malignos en Cuba [Tesis]. *La Habana*, 2009.

[2] Organización Panamericana de la Salud. Plan de Acción Regional de Prevención y Control del Cáncer, Junio 2008. Disponible en: http://www1.paho.org/Spanish/AD/DPC/NC/pcc-stakeholders-08.htm.

[3] MINSAP. Anuario Estadístico Nacional 2013, Enero 2014.

[4] Anuario Estadístico de salud en Cuba - 2014, 2014. Disponible en: http://files.sld.cu/bvscuba/files/2015/04/anuario-estadistico-de-salud-2014.pdf.

[5] INOR. Instituto Nacional de Oncología y Radiología de Cuba.Registro Nacional del Cáncer. Incidencia por Cáncer en Cuba, marzo 2010.

[6] Fernández-Mayoralas, G. Una aproximación a la epidemiología del SIDA en España. In *Habitar, vivir, prever: actas del V Congreso de la Población Española*, pages 257–265. Asociación de Geógrafos Españoles, 1995.

[7] for Research on Cancer, I. A. et al. *IARC handbooks of cancer prevention*, volume 10. IARC, 2005.

[8] Group, U. C. S. W. et al. United States cancer statistics: 1999–2010 incidence and mortality web-based report. *Atlanta: US Department of Health and Human Services, Centers for Disease Control and Prevention and National Cancer Institute*, 2013.

[9] Oliva, L. Cáncer y Ambiente. *Bases Epidemiológicas para su Investigación y Control, Centro Panamericano de Ecología Humana y Salud, Programa de Salud Ambiental, OPS-OMS, México*, 1990.

[10] Georreferenciación y sistemas de coordenadas | ArcGIS Resource Center. Disponible en: http://resources.arcgis.com/es/help/getting-started/articles/026n0000000s000000.htm.

[11] Bravo, J. D. *Breve introducción a la cartografía ya los sistemas de información geográfica (SIG)*. Ciemat, 2000.

[12] CanquiLlusco, J. E. Georreferenciación. *Revista de Información, Tecnología y Sociedad*, page 22, 2012.

[13] De Pietri, D. E.; García, S. & Rico, O. Modelos geo-espaciales para la vigilancia local de la salud. 2008. Disponible en: http://iris.paho.org/xmlui/handle/123456789/9975.

[14] Pfeiffer, D.; Robinson, T.; Stevenson, M.; Stevens, K. B.; Rogers, D. J. & Clements, A. C. Spatial analysis in epidemiology. *Healthcare Informatics Research*, 19(2):148, 2013. Disponible en: http://synapse.koreamed.org/pdf/10.4258/hir.2013.19.2.148, doi:10.4258/hir.2013.19.2.148.

[15] Rodríguez Corvea, L. *Detención de conglomerados en la solución de problemas bioinformáticos y biomédicos*. PhD thesis, Universidad Central "Marta Abreu" de las Villas. Facultad de Matemática, Física y Computación, 2011.

[16] Duda, R. O.; Hart, P. E. et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[17] Duda, R. O.; Hart, P. E. & Stork, D. G. *Pattern classification*. John Wiley & Sons, 2012.

[18] Dempster, A. P.; Laird, N. M. & Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[19] Fukunaga, K. *Introduction to statistical pattern recognition*. Academic press, 2013.

[20] Jain, A. K.; Murty, M. N. & Flynn, P. J. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[21] Jain, A. K. & Flynn, P. J. *Image segmentation using clustering*. IEEE Press, Piscataway, NJ, 1996.

[22] Jain, A. K.; Duin, R. P. & Mao, J. Statistical pattern recognition: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4–37, 2000.

[23] Salton, G. Automatic term class construction using relevance—a summary of work in automatic pseudoclassification. *Information Processing & Management*, 16(1):1–15, 1980.

[24] Zhai, C. & Lafferty, J. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.

[25] Cutting, D. R.; Karger, D. R.; Pedersen, J. O. & Tukey, J. W. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM, 1992.

[26] Dhillon, I. S.; Mallela, S. & Kumar, R. Enhanced word clustering for hierarchical text classification. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 191–200. ACM, 2002.

[27] Xu, X.; Ester, M.; Kriegel, H.-P. & Sander, J. A distribution-based clustering algorithm for mining in large spatial databases. In *Data Engineering, 1998. Proceedings., 14th International Conference on*, pages 324–331. IEEE, 1998.

[28] Ester, M.; Frommelt, A.; Kriegel, H.-P. & Sander, J. Spatial data mining: database primitives, algorithms and efficient DBMS support. *Data Mining and Knowledge Discovery*, 4(2-3):193–216, 2000.

[29] González Polanco, L. & Pérez Betancourt, G. La minería de datos espaciales y su aplicación en los estudios de salud y epidemiología. *Revista Cubana de Información en Ciencias de la Salud*, 24(4):482–489, 2013. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2307-21132013000400010.

[30] Cades, I.; Smyth, P. & Mannila, H. Probabilistic modeling of transactional data with applications to profiling, visualization and prediction, sigmod. *Proc. of the 7th ACM SIGKDD. San Francisco: ACM Press*, pages 37–46, 2001.

[31] Heer, J. & Chi, E. H. Identification of web user traffic composition using multi-modal clustering and information scent. In *Proc. of the Workshop on Web Mining, SIAM Conference on Data Mining*, pages 51–58, 2001.

[32] Foss, A.; Wang, W. & Zaïane, O. R. A non-parametric approach to web log analysis. In *Proc. of Workshop on Web Mining in First International SIAM Conference on Data Mining*, pages 41–50. Citeseer, 2001.

[33] Xu, Y.; Olman, V. & Xu, D. Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. *Bioinformatics*, 18(4):536–545, 2002.

[34] González, D. P. *Algoritmos de agrupamiento basados en densidad y variación de clusters*. PhD thesis, Universitat Jaume I, Departament de Llenguatges i Sistemes Informàtics, 2010. Disponible en: http://www.cerpamid.co.cu/sitio/files/DamarisTesis.pdf.

[35] García, A. O. & Reyes, A. J. O. Análisis de algoritmos y sistemas de información electrónica para potenciar la toma de decisiones clínicas en el SIAPS. *Serie Científica*, 6(7), 2013. Disponible en: https://publicaciones.uci.cu/index.php/SC/article/view/1130.

[36] Pérez, J.; Henriques, M.; Pazos, R.; Cruz, L.; Reyes, G.; Salinas, J. & Mexicano, A. Mejora al algoritmo de agrupamiento K-means mediante un nuevo criterio de convergencia y su

aplicación a bases de datos poblacionales de cáncer. *II Taller Latino Iberoamericano de Investigación de Operaciones*, 2007. Disponible en: https://www.researchgate.net/profile/Laura_Cruz_Reyes/publication/228882107_Mejora_al_algoritmo_de_agrupamiento_K-means_mediante_un_nuevo_criterio_de_convergencia_y_su_aplicacion_a_bases_de_datos_poblacionales_de_cancer/links/53d3dd870cf220632f3ce8b7.pdf.

[37] MacQueen, J. et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[38] HARTIGAN, J. & WONG, M. A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[39] Chen, C. W.; Luo, J. & Parker, K. J. Image segmentation via adaptive K-mean clustering and knowledge-based morphological operations with biomedical applications. *Image Processing, IEEE Transactions on*, 7(12):1673–1683, 1998.

[40] Ester, M.; Kriegel, H.-P.; Sander, J. & Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[41] Vázquez, F.; Sánchez, J. S. & Pla, F. A stochastic approach to Wilson's editing algorithm. In *Pattern Recognition and Image Analysis*, pages 35–42. Springer, 2005.

[42] Mesa, F. D. V. *Algoritmos de aprendizaje continuo mediante selección de prototipos para clasificadores basados*. PhD thesis, Universitat Jaume I, 2008.

[43] Tran, T. N.; Wehrens, R. & Buydens, L. M. Knn density-based clustering for high dimensional multispectral images. In *Remote Sensing and Data Fusion over Urban Areas, 2003. 2nd GRSS/ISPRS Joint Workshop on*, pages 147–151. IEEE, 2003.

[44] Bishop, C. M. Pattern Recognition. *Machine Learning*, 2006. Disponible en: www.springer.com/cda/content/document/cda_downloaddocument/9780387310732-c1.pdf?SGWID=0-0-45-284415-p134256227.

[45] Sánchez Navarro, R. SECUENCIAS Y SUBSECUENCIAS DE PASES ENTRE FUTBOLISTAS ¿ES XAVI UN JUGADOR UNICO? 2015. Disponible en: http://masteres.ugr.es/moea/pages/curso201415/tfm1415/sancheznavarro_tfm/!

[46] Aronoff, S. *Geographical Information Systems: A management perspective*. WDL Publications, Otawa Canadá, 1989.

[47] Delgado, T. Infraestructura cubana de datos geoespaciales: una necesidad nacional para la integración y diseminación de datos geoespaciales. In *Memorias del II Congreso Internacional Geomática 2000*, 2000.

[48] Bravo, J. D. *Breve introducción a la cartografía ya los sistemas de información geográfica (SIG)*. Ciemat, 2000.

[49] Lorenzo, R. Cartografía: Urbanismo y desarrollo inmobiliario. *Madrid, Cie Inversiones Editoriales Dossat*, 2000.

[50] Map production International Cartographic Association. Disponible en: http://icaci.org/research-agenda/map-production/.

[51] Torres, A. R. & Puente, R. R. Servicio de mapas temáticos. *Mapping*, (139):36–39, 2010.

[52] Víctor, O. Sistemas de Información Geográfica. *Libro SIG*, 2011.

[53] GRASS, G. The world's leading Free GIS software'. *URL: http://grass. osgeo. org*, 2013.

[54] Böhner, J.; Conrad, O.; Köthe, R. & Ringeler, A. System for automated geoscientific analyses. Disponible en: http://www.saga-gis.org/en/index.html.

[55] QGIS, A. Free and Open Source Geographic Information System. Disponible en: http://www.qgis.org/en/site/.

[56] Schefer-Wenzl, S.; Sobernig, S. & Strembeck, M. Evaluating A Uml-Based Modeling Framework For Process-Related Security Properties: A Qualitative Multi-Method

Study. In *ECIS*, page 134. Disponible en: http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1357&context=ecis2013_cr.

[57] HERRAMIENTAS CASE de miguel angel gonzalez en Prezi. Disponible en: https://prezi.com/aad8mbta_vjb/herramientas-case/.

[58] Software Design Tools for Agile Teams, with UML, BPMN and More. Disponible en: https://www.visual-paradigm.com/.

[59] LOUDEN, K. C. *Lenguajes de programación: Principios y práctica*. Cengage Learning Latin America, 2004.

[60] Duque, R. G. Python para todos. Disponible en: mundogeek.net/tutorial-python/.

[61] scikit-learn: machine learning in Python — scikit-learn 0.17.1 documentation. Disponible en: http://scikit-learn.org/stable/.

[62] PyCharm. Intelligent Python IDE with refactorings, debugger, code completion, on-the-fly code analysis and coding productivity orientation . Disponible en: https://www.jetbrains.com/pycharm/.

[63] Cobo, Á. *Diseño y programación de bases de datos*. Disponible en: http://books.google.es/books?hl=es&lr=&id=anCDr9N-kGsC&oi=fnd&pg=PA7&dq=Dise%C3%B1o+y+programaci%C3%B3n+de+bases+de+datos&ots=UXEBp8mpzV&sig=jPWxCyBUit3XHIQIr4NpzhIbUwQ.

[64] PostgreSQL: Documentation: 9.0: Release 9.0.1. Disponible en: https://www.postgresql.org/docs/9.0/static/release-9-0-1.html.

[65] PostGIS 2.0 Manual, 2014. Disponible en: http://postgis.net/docs/manual-2.0/.

[66] Robinson, C. Basic introduction into pgAdmin III and SQL queries. Disponible en: https://library.thehumanjourney.net/658/.

[67] Letelier, P. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Disponible en: www.cyta.com.ar/ta0502/b_v5n2a1.htm.

[68] Joskowicz, J. Reglas y prácticas en eXtreme Programming. *Universidad de Vigo*. Disponible en: http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf.

[69] Beck, K. *Extreme programming explained: embrace change*. Disponible en: https://books.google.es/books?hl=es&lr=&id=G8EL4H4vf7UC&oi=fnd&pg=PR13&dq=Extreme+programming+explained:+embrace+change+[online].+Addison-Wesley+Professional.+&ots=j9yIrujYxk&sig=8XY-aKnAt1OXxjtMi8nrIj0QwQ8#v=onepage&q=Extreme%20programming%20explained%3A%20embrace%20change%20[online].%20Addison-Wesley%20Professional.&f=false.

[70] Sommerville, I. & Galipienso, M. I. A. *Ingeniería del software*. Disponible en: https://books.google.es/books?hl=es&lr=&id=gQWd49zSut4C&oi=fnd&pg=PA1&dq=Ingenier%C3%ADa+del+software.+Pearson+Educaci%C3%B3n.&ots=s656ptwvtf&sig=rNlnqw86hZOeG7oXJPvvZJzJEo8#v=onepage&q=Ingenier%C3%ADa%20del%20software.%20Pearson%20Educaci%C3%B3n.&f=false.

[71] Casas, S. & Reinaga, H. Identificación y modelado de aspectos tempranos dirigido por tarjetas de responsabilidades y colaboraciones. In *XIV Congreso Argentino de Ciencias de la Computación*. Disponible en: http://sedici.unlp.edu.ar/handle/10915/21813.

[72] Pressman, R. Ingeniería del software. Un enfoque práctico. Sexta edición. Editoria l McGraw-Hill, 2005.

[73] Larman, C. UML y patrones. introducción al análisis y diseño orientado a objetos. 1999.

[74] Grey, L. W. G. & Viltres, Y. M. Proceso de mejora del Sistema de Gestión de Proyectos para Cuba y Venezuela. *Campus Virtuales*, 3(2):16–22. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=5166889.

[75] IDERC. Infraestructura de Datos Espaciales de la República de Cuba | IDERC. Disponible en: http://www.iderc.cu/web/iderc/inicio.

[76] Companioni, Y. B. Construcción de tipologías: metodología de análisis para la estratificación según indicadores de salud. *Reporte Técnico de Vigilancia*, 2005. Disponible en: http://www.sld.cu/galerias/pdf/sitios/vigilancia/bombino.pdf.

GLOSARIO DE TÉRMINOS

GRASS Geographic Resources Analysis Support System

SAGA Sistema de Análisis Geocientífico Automatizado (del inglés System for Automated Geoscientific Analyses)

Qgis Quantum GIS

SIG Sistemas de Información Geográfica (del inglés Geographic Information Systems)

CASE Herramienta que brinda asistencia a los analistas (del inglés Computer Aided Software Engineering)

CRC Clase, Responsabilidad y Colaboración (del inglés Class, Responsability and Collaboration)

GoF Patrones del Grupo de Cuatro (del inglés Gang of Four)

GRASP Patrones Generales de Software para Asignación de Responsabilidades (del inglés General Responsibility Assignment Software Patterns)

HU Historia de Usuario (artefacto generado por la metodología Programación Extrema)

RF Requisito Funcional

RNF Requisito No Funcional

XP Metodología de desarrollo de software ágil (del inglés Extreme Programing)