UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Componente para la geo-simulación basada en autómatas celulares sobre el Sistema de Información Geográfica QGis

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

Autores: Claudia Guerra Fernández

Miguel Autberto Jiménez Benzol

Tutores: Ing. Liset González Polanco

Ing. Yadian Guillermo Pérez Betancourt

La Habana, 20 de junio de 2017

Declaración de autoría

Declaramos ser los autores 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 2017.

Claudia Guerra Fernández	Miguel Autberto Jiménez Benzol
Ing. Liset González Polanco	Ing. Yadian Guillermo Pérez Betancourt

Dedicatoria

A mi abuelo Raúl Manuel Quintana Saínz, que hace 2 años el cáncer lo arrebato de mi lado, el cual sé que estaría muy orgulloso de verme convertida en ingeniera.

A mi mamá Lídice Fernández Calcagno.

A mi abuela María Josefa Calcagno Gascón.

A mi hermana Lídice Guerra Fernández.

Claudia Guerra Fernández

A mi abuela Fransica Lucia Roa Cedeño.

Miguel Autberto Jiménez Benzol

Resumen

Los Sistemas de Información Geográfica (SIG) se han convertido en herramientas básicas dentro del análisis, procesamiento y transformación de la información almacenada en bases de datos espaciales; brindan un punto de partida para la organización y actualización de la información espacial que manejan entidades gubernamentales o no. Su empleo en estudios sobre la distribución espacial de problemas de salud basado en simulaciones es limitado, sobre todo por las insuficiencias y dispersión de las herramientas existentes para la incorporación de la componente espacial. En este contexto, la incorporación de los SIG es aún insuficiente, reconociéndose que en Cuba las distribución espacial de problemas de salud ha sido poco estudiada. El objetivo del presente trabajo es desarrollar un componente para la geo-simulación basada en autómatas celulares sobre el Sistema de Información Geográfica QGis que facilite la obtención de información prospectiva. Para ello se efectuó un estudio del panorama actual de los softwares SIG y se realizó un análisis de las herramientas existentes que soportan los procesos de geo-simulación. Se obtuvo un sistema que permite integrar datos de variada naturaleza para la obtención de información prospectiva; la solución tiene su base en los métodos de simulación para realizar análisis prospectivo, contribuye a la identificación de riesgos de salud en territorios y a la toma de decisiones en las entidades de salud.

Palabras clave: distribución espacial, geo-simulación, autómatas celulares, información prospectiva.

Abstract

Geographic Information Systems (GIS) have become basic tools in the analysis, processing and transformation of information stored in spatial databases; provide a starting point for the organization and updating of spatial information handled by governmental entities or not. Its use in studies on the spatial distribution of health problems based on simulations is limited, mainly due to the inadequacies and dispersion of existing tools for the incorporation of the space component. In this context, the incorporation of GIS is still insufficient, recognizing that in Cuba the spatial distribution of health problems has been little studied. The objective of the present work is to develop a component for the geo-simulation based on cellular automata on the Geographic Information System QGis that facilitates the obtaining of prospective information. For this, a study of the current panorama of the GIS softwares was made and an analysis of the existing tools that support the processes of geo-simulation was realized. A system was obtained that allows the integration of data of varied nature for obtaining prospective information; the solution is based on simulation methods to perform prospective analysis, contributes to the identification of health risks in territories and to decision making in health institutions.

Keywords: spatial distribution, geo-simulation, cellular automata, prospective information.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En el transcurso de los años se ha evidenciado un avance de la tecnología, tributando a la evolución y transformación de algunas de las esferas de la sociedad, por ejemplo: la agricultura, la meteorología, el turismo y la medicina. Una herramienta muy popular y de gran impacto en los últimos tiempos para realizar estudios asociados a estas esferas son los Sistemas de Información Geográfica (SIG).

Un SIG se puede considerar como la unión de una serie de componentes físicos y lógicos que permiten la gestión de información geográficamente referenciada. La ventaja del uso de los SIG está dada por la representación gráfica que se puede realizar de la información almacenada en bases de datos, que puede ser comparada y visualizada en mapas, con un "plus" de dinamismo como las escalas y los cálculos geográficos [1].

Estos sistemas se han convertido en unas herramientas básicas dentro del análisis, procesamiento y transformación de la información en bases de datos espaciales que puede ser utilizada y supervisada por personal experto; sirve como punto de partida para la organización y actualización de la información que manejan las entidades prestadoras del servicio de salud [2].

Entre los antecedentes de aplicación SIG sobre datos de salud, se destaca la representación gráfica de la distribución espacial de las enfermedades, de gran interés para mostrar geográficamente las tasas de incidencia con objetivos puramente descriptivos, y de mortalidad. La representación espacial también ha sido utilizada para formular hipótesis relacionadas con la etiología[1] de enfermedades y/o documentar o establecer el marco de estudios de la epidemiología [3].

El análisis de la distribución espacial de enfermedades ha aumentado de forma considerable, plasmando una forma de analizar y sistematizar una problemática en crecimiento desde el punto de vista integral, contextualizando las características que pueden influir en el aumento o la distribución de alguna enfermedad con el empleo de la simulación geográfica. Esta actividad consiste en la imitación del funcionamiento de

[1] Etiología: parte de la medicina que estudia el origen o las causas de las enfermedades.

un sistema real durante un intervalo de tiempo, realizándose de forma manual o computacional. Se basa en un modelo de la realidad que describe un suceso y al observar el comportamiento de este, permite obtener conocimiento acerca del sistema real.

El modelado ha consistido tradicionalmente en plantear y resolver ecuaciones que describen el fenómeno en estudio. En la realidad, existen sistemas que no pueden ser representados en términos de ecuaciones; las simulaciones computacionales son la herramienta adecuada para el estudio de este tipo de sistemas complejos[4].

El empleo de los SIG en estudios sobre la distribución espacial de problemas de salud basado en simulaciones es limitado, sobre todo por las insuficiencias y dispersión de las herramientas existentes para la incorporación de la componente espacial. Los problemas de modelación de sistemas complejos se realizan con ecuaciones diferenciales, lo que limita la integración del espacio en la simulación. En este contexto los SIG constituyen herramientas de análisis de mucho valor, pero su incorporación en este tipo de estudios es aún insuficiente, motivado por: ***i)*** acceso limitado a los SIG por los costos que ellos implican, ***ii)*** poco conocimiento de las herramientas, ***iii)*** el tiempo de formación en el área de los SIG es elevado y iv) modelos de geo-simulación limitados. En Cuba se ha reconocido además que las distribuciones espaciales de los problemas de salud, ha sido poco estudiada [5].

Por lo antes expuesto se plantea el siguiente ***problema a resolver:*** las limitaciones en los Sistemas de Información Geográfica para realizar estudios basados en la geo-simulación dificultan la obtención de información prospectiva, identificando como ***objeto de estudio:*** Modelos de geo-simulación, enmarcado en el ***campo de acción:*** geo-simulación de la distribución espacial sobre problemas de la salud. En aras de resolver el problema planteado se tiene el siguiente ***objetivo general:*** desarrollar un componente para la geo-simulación basada en autómatas celulares sobre el Sistema de Información Geográfica QGis que facilite la obtención de información prospectiva.

Basándose en el objetivo general se formulan los siguientes ***objetivo específicos:***

1. Construir el marco teórico referencial de la investigación, relacionado con los modelos de Geo-simulación.
2. Diseñar las estructuras de datos y algoritmos relacionados con los autómatas celulares.
3. Implementar un complemento para QGis que de soporte a la geo-simulación en autómatas celulares.

4. Verificar la solución informática propuesta aplicando diferentes pruebas y métricas.

Dada la problemática se plantea la siguiente ***Idea a defender:***
Si se desarrolla un componente para la geo-simulación basada en autómatas celulares, entonces se facilitará la obtención de información prospectiva para la planificación de recursos y la conformación de hipótesis.

Como ***posibles resultados:***
Obtener un componente para la geo-simulación basada en autómatas celulares sobre el Sistema de Información Geográfica QGis que facilite la obtención de información prospectiva para la planificación de recursos y la conformación de hipótesis.

En el desarrollo del presente trabajo de investigación fueron utilizados los siguientes **métodos científicos** [6]:

Métodos teóricos:

- **Histórico-Lógico:** permitió realizar un estudio de las principales herramientas que realizan procesos de geo-simulación basada en autómatas celulares.
- **Analítico -sintético:** se utiliza para identificar y analizar las diversas funcionalidades de los SIG que pueden ser aplicadas al proceso de geo-simulación basada en autómatas celulares y su posterior síntesis, conforme a las necesidades de Cuba en el sector de la salud.
- **Modelación:** se emplea para mostrar los diversos diagramas que se construyen como resultado del proceso de ingeniería de software.

Métodos empíricos:

- **Análisis documental:** en la revisión de la literatura especializada para extraer la información necesaria que permitió realizar el proceso de investigación.
- **Análisis comparativo:** para detectar similitudes, diferencias e insuficiencias en cuanto a la interpretación que brindan las herramientas existentes para modelar los procesos.
- **Observación:** para adquirir conocimiento sobre el campo de acción a través de la investigación realizada sobre las herramientas.
- **Experimentación:** se utilizó para la demostración de los resultados obtenidos y la validación de los mismos, a partir de datos reales, suministrados por el Anuario Estadístico del año 2001 [7].

Estructura del trabajo:

El presente trabajo está estructurado de la siguiente forma: introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, glosario de términos y anexos. A continuación, se muestra una breve descripción de cada uno de los capítulos.

Capítulo 1: Referentes teóricos sobre la geo-simulación basadas en autómatas celulares

En este capítulo se describen los fundamentos teóricos que constituyen las bases sobre las que se realiza esta investigación. Se presentan una serie de conceptos asociados a Sistemas de Información Geográfica como soporte para la geo-simulación basados en autómatas celulares. Además, se fundamentan las tecnologías, lenguajes y herramientas de desarrollo a utilizar.

Capítulo 2: Componente informático para la geo-simulación basada en autómatas celulares sobre el Sistema de Información Geográfica QGis

En este capítulo se describen los elementos relacionados con el desarrollo de un componente informático para la geo-simulación basada en autómatas celulares sobre el Sistema de Información Geográfica QGis. 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 historia de usuario (HU) definida en la fase de planificación.

Capítulo 3: Implementación y validación de la solución propuesta.

En el capítulo se ejecutarán las pruebas y validaciones del algoritmo las cuales constituyen un instrumento para comprobar el nivel de calidad de un producto. La metodología divide las pruebas en dos grupos: pruebas unitarias, encargadas de verificar el código, la cual es realizada por los programadores, como también plantea las pruebas de aceptación destinadas a evaluar si al terminar una iteración se consiguió la funcionalidad requerida diseñadas por el cliente. Se establece el estándar de codificación que se utilizará en el desarrollo de la solución. Durante las iteraciones las historias de usuarios seleccionadas serán traducidas a pruebas de aceptación. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una historia de usuario ha sido implementada correctamente.

Capítulo 1

REFERENTES TEÓRICOS SOBRE LA GEO-SIMULACIÓN Y LA SIMULACIÓN BASADAS EN AUTÓMATAS CELULARES.

En este capítulo se describen los fundamentos teóricos que constituyen las bases sobre las que se realiza esta investigación. Se presentan una serie de conceptos asociados a Sistemas de Información Geográfica como soporte para la geo-simulación basados en autómatas celulares. Además, se fundamentan las tecnologías, lenguajes y herramientas de desarrollo a utilizar.

1.1. Conceptos asociados al dominio del problema

La geo-simulación basado en autómatas celulares tiene asociada un conjunto de definiciones que conforman el marco teórico referencial para el desarrollo de la investigación.

1.1.1. Cartografía

Cartografía es la disciplina que se ocupa de la concepción, producción, difusión y estudio de los mapas. 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 [8].

1.1.2. Datos espaciales

Los datos espaciales representan información sobre la ubicación física en forma de objetos geométricos, estos objetos pueden ser ubicaciones de puntos u objetos más complejos como países, carreteras o lagos.

Un dato espacial es una variable asociada a una localización del espacio, normalmente se utilizan datos vectoriales, los cuales pueden ser expresados mediante tres tipos de objetos espaciales: puntos, líneas y polígonos [9].

1.1.3. Mapa

Un mapa es cualquier tipo de representación geográfica de algún territorio, en una superficie plana, bidimensional, tridimensional o esférica [10].

1.1.4. Mapas topográficos

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 [10].

1.1.5. Mapas temáticos

Es un mapa topográfico que demuestra cualquier fenómeno de la superficie terrestre en diferentes escalas, los mismos pueden tratar sobres los aspectos más relevantes de un espacio geográficos, por ejemplo: los ríos más importantes de América. El mapa temático es mostrado a través de una ilustración sencilla, fácil de comprender para el lector [10].

1.1.6. Información prospectiva

El uso de modelos predictivos para la generación de escenarios futuros, ya sea en el contexto de la planificación territorial o en la evaluación de impacto ambiental, representa una importante oportunidad para anticipar, prevenir y mitigar dinámicas insostenibles[11].

La prospección o análisis prospectivo, es un método de planificación que permite la identificación de un futuro probable y de un futuro deseable, a partir de la acción presente, permitiendo ampliar la base de información disponible para la toma de decisiones [12].

El análisis prospectivo considera tres tipos de futuro: el futuro tendencial, el futuro exploratorio y el futuro normativo.

El futuro tendencial está determinado por una única tendencia mostrada en el pasado considerando cierta

desviación (ver Figura 1.1), para este análisis se utilizan técnicas de previsión o métodos extrapolativos, tales como proyección simple, curvas S, y analogía histórica [13].

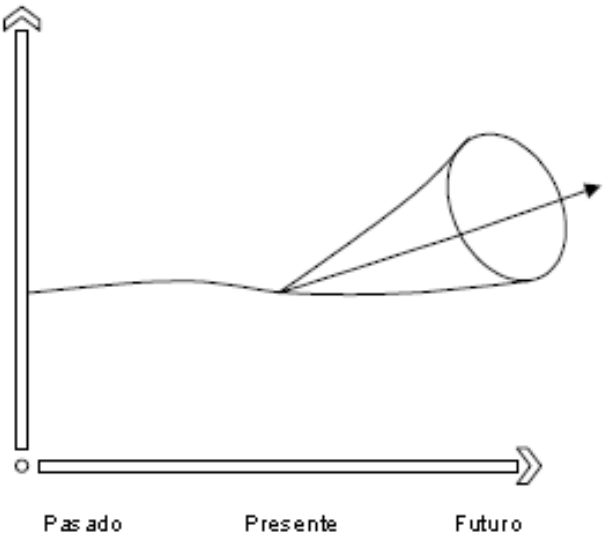

Figura 1.1: *Futuro único y verdadero: El futuro es parecido al pasado.*

Fuente: *(Gomes, 2001)*

El futuro exploratorio considera varias posibilidades de futuro construidas por expertos (ver Figura1.2), lo cual permite evaluar cambios direccionales y la complejidad de los fenómenos, algunas de las técnicas utilizadas son: Delphi, análisis morfológico y construcción de modelos [13].

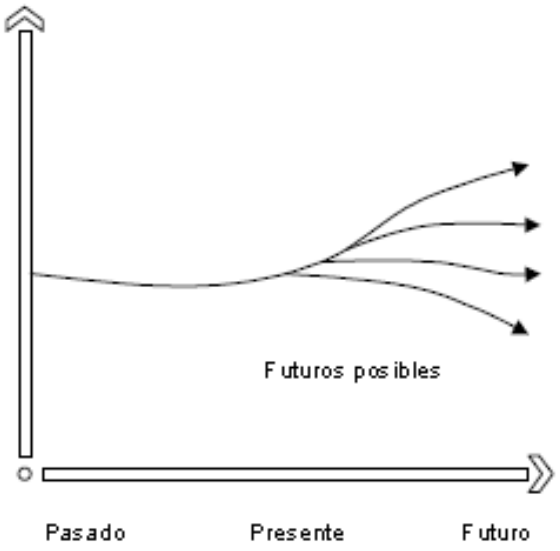

Figura 1.2: *Futuro múltiple: El futuro puede ser diferente.*

Fuente: *(Gomes, 2001)*

El futuro normativo es múltiple e incierto, mediante su uso se puede diseñar una imagen de un futuro lógico a través de técnicas exploratorias con uso normativo (ver Figura1.3).[13]

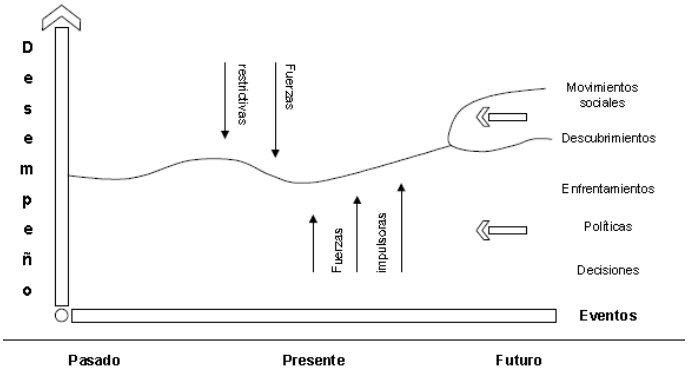

Figura 1.3: *El futuro múltiple e incierto.*

Fuente: *(Gomes, 2001)*

Desde la perspectiva del futuro normativo aparecen tres conceptos fundamentales:

- **Factor crítico:** es cualquier variable o estructura (conjunto de variables) de alto impacto que afecta, positiva o negativamente, el desempeño de un sistema.

- **Fuerza impulsora:** es cualquier fenómeno, variable o estructura que afecta de forma positiva el desempeño de un factor crítico.
- **Fuerza restrictiva:** es cualquier fenómeno, variable o estructura que afecta de forma negativa el desempeño de un factor crítico.

En este enfoque se plantea esencialmente que el futuro puede ser construido por la sociedad utilizando la información disponible para la toma de decisiones, comprendiendo el futuro como el resultado de la interacción entre tendencias históricas y los eventos hipotéticos [13].

Es importante resaltar que, más que tratar de prever el futuro, la prospección busca orientar la toma de decisiones en la actualidad, partiendo de las tendencias de comportamiento de las variables, de manera que los escenarios solo son construcciones hipotéticas de eventos futuros que permiten construir una imagen de futuros alternativos, reducir la complejidad y la incertidumbre [13].

En la Tabla1.1 se evidencian los tres tipos de futuros que considera la información prospectiva. Los autores definen que se utiliza el normativo que permite diseñar un futuro deseable y una imagen del futuro lógico mediante las técnicas exploratorias con uso normativo. Esto permite responder a las preguntas: ¿Cuál es el futuro que se desea? ¿Cómo podría ser el futuro si se continúa en una dirección similar a la actual?

Tabla 1.1: *Tabla comparativa entre los tipos de futuros.*

Fuente: *(Elaboración propia)*

Futuros	Única tendencia	Toma de decisiones	Imagen de futuros alternativos	Evalúa cambios direccionales	Complejidad de los fenómenos
Tendencial	x	x			
Exploratorio		x		x	x
Normativo		x	x	x	x

Continuamente se es testigo de conflictos y contradicciones como resultado del enfrentamiento dialéctico entre diferentes visiones del mundo. La ciencia moderna originada en los siglos XVI y XVII propone una visión mecánica del mundo haciendo una metáfora con una máquina racional, sin embargo, con la emergencia de la revolución en torno a la tecnología de la información esta máquina gana una dimensión cibernética [12].

- **Visión mecánica:** el futuro está asociado a la idea del progreso técnico de forma que puede ser

previsto, planificado y controlado.

- **Visión holística:** el futuro no puede ser previsto, puede ser comprendido e influenciado.

Desde la visión holística el mundo es un sistema complejo, multidimensional e interdependiente, cuya dinámica incluye múltiples funciones, conflictos y contradicciones. La visión holística lleva a la necesidad de un futuro múltiple, incierto, dinámico y construido socialmente, en el cual lo importante es conocer las influencias que forjan sus tendencias para negociar intereses, superar conflictos y contradicciones.

Métodos para el análisis prospectivo

A continuación, se presenta una breve descripción de los principales métodos disponibles para realizar el análisis prospectivo[12].

Ábaco de Regnier

Es una técnica en la que expertos exponen su opinión frente a un tema determinado para que posteriormente sean sometidos a votación, se aplica principalmente para estimar el comportamiento de un grupo de factores y determinar la intensidad de un problema.

El Ábaco de Regnier es probablemente la más versátil y sencilla de las técnicas prospectivas, pudiéndose utilizar también para calificar y clasificar problemas y servir de base para procesos de planificación [13].

El método del Análisis Estructural

Este método es aplicado con la participación de un pequeño grupo de expertos cuya misión es identificar y clasificar las variables claves de un sistema cuyos componentes están íntimamente relacionados, permitiendo identificar las variables que conforman la problemática, así como todas las relaciones que pueden tener entre sí.

Las etapas que se deben cubrir para la realización del ejercicio son: a) identificación de las variables que componen el problema, b) detección de la influencia que ejercen entre sí, y c) determinación de las variables más sobresalientes. El éxito en su aplicación reside en la disponibilidad de información para la construcción de escenarios [13].

El método del Juego de Actores

Este método es desarrollado por un grupo de expertos que buscan identificar las alianzas y conflictos que surgen en relación con un tema específico, así como los proyectos, anhelos y temores que pueden indicar la evolución o potencialidad de los problemas en un sistema.

Como punto de partida se tienen los problemas identificados en el análisis estructural, los cuales se complementan con información sobre el estado actual del problema, la proyección de algún dato útil y las tendencias internacionales respecto del problema en estudio [13].

Matriz del impacto cruzado

La filosofía básica de la Matriz del Impacto Cruzado es que ningún evento se realiza aislado, sino más bien influenciado, con mayor o menor probabilidad, por la ocurrencia de otros eventos. El impacto cruzado intenta captar estas interrelaciones a partir de juicios estimativos de expertos. En esta metodología los expertos comunican su opinión mediante probabilidades matemáticas, asignando valores a la ocurrencia de determinado tipo de eventos [13].

Método Delphi

Es un método orientado a agrupar y refinar los juicios emitidos por grupos de expertos con respecto a varias opciones que se desean evaluar, para lo cual se utilizan medidas estadísticas. Este método es altamente útil cuando no se dispone de datos cuantitativos, o cuando se requiere contar con un enfoque interdisciplinario. Para su aplicación se requiere la participación de un gran número de personas, lo cual a su vez implica que las posiciones minoritarias son suprimidas [13].

La construcción de escenarios

Un escenario es la descripción particular de un futuro posible y del proceso de acercamiento a este futuro. El objetivo de esta técnica es establecer futuros alternativos soportados con datos históricos de referencia. Esta técnica permite identificar los límites del grado de incertidumbre de los elementos claves para la toma de decisiones, así como generar tendencias que son acotadas por rangos [13].

Modelos de simulación

Un modelo es una abstracción de la realidad compuesta por componentes interactivos, que trata

de interpretar y proyectar una condición existente, así como reflejar relaciones entre las variables interdependientes. Los modelos de simulación son particularmente útiles como instrumentos para analizar las probables consecuencias bajo diversos escenarios.

Los modelos matemáticos son herramientas que formalizan de manera más precisa los modelos mentales, aunque no necesariamente son superiores a los modelos mentales; entre las ventajas de los modelos matemáticos se pueden mencionar las siguientes: son explícitos, pueden ser examinados minuciosamente y su ejecución en computadora posibilita una visión ampliada de la complejidad de los sistemas.

La simulación matemática presenta la desventaja de ser costosa por el tiempo de desarrollo y validación del modelo. Además, los modelos suelen ser rígidos y no adaptables a los cambios, de forma que es necesario resaltar que la visión de futuro debe ser constantemente actualizada, pues el contexto cambiante no permite mantener visiones fijas. Esto no significa cambiar de visión frecuentemente, significa tener los medios para saber cuándo la visión de futuro necesita ser ajustada, o incluso, cuando debe ser reemplazada [13].

Comparación de métodos de prospección

A partir de los datos de la Tabla 1.2 se evidencia que los distintos métodos de prospección son complementarios, algunos tienen aplicación en la identificación de variables e interrelaciones, otros en su organización jerárquica y otros en la construcción de escenarios. La mayoría de los métodos descritos dependen de la opinión de expertos en la formulación de escenarios y en su proyección.

De manera, que aunque en la formulación de escenarios, la identificación de variables y de interrelaciones se puedan utilizar métodos como la matriz de impacto cruzado, el método de análisis estructural y el juego de actores, para la evolución y prospección de variables en tiempo y espacio, resulta de mucha utilidad la utilización de modelos de simulación como los autómatas celulares que permiten construir futuros alternativos inciertos y dinámicos, desde una visión holística, utilizando técnicas exploratorias con uso normativo.

Tabla 1.2: *Tabla comparativa de las métodos para el análisis prospectivo.*

Fuente: *(Elaboración propia)*

Metodología	Formulación de escenarios	Identificación de variables	Organización jerárquica	Opinión de expertos	Evolución y prospección de variables en tiempo
Ábaco de Regnier	X			X	
El método del Análisis Estructural	X	X		X	
El método del Juego de Actores		X		X	
Matriz del Impacto Cruzado	X	X		X	
Método Delphi		X		X	
La construcción de escenarios	X	X			
Modelos de simulación	X	X	X	X	X

1.1.7. Simulación

La simulación es la imitación del funcionamiento de un sistema real durante un intervalo de tiempo. Esta simulación puede realizarse ya sea de forma manual o computacional. La simulación se basa en un modelo de la realidad que cuenta una historia y permite obtener conocimiento acerca del sistema real. El comportamiento de la simulación está determinado por un conjunto de supuestos concernientes al sistema real, estos se expresan a través de relaciones lógicas y matemáticas entre las entidades [14].

1.1.8. Geo-simulación y autómatas celulares (AC)

Los sistemas geográficos están determinados por el comportamiento humano, las decisiones de los seres humanos, la unidad del sistema y su dinámica

Entender la dinámica del sistema como un fenómeno colectivo, ha implicado un cambio de paradigma, pasando de un enfoque reduccionista a uno generativo. En estos sistemas emergentes, un pequeño número de reglas o leyes, aplicadas a nivel local y entre muchas entidades, son capaces de generar complejos comportamientos colectivos, de tal manera que las acciones de las partes no se limitan a la actividad del conjunto.

En los sistemas espaciales dinámicos, todos los objetos cambian sus propiedades y/o situación; y la meta de un modelo geográfico es imitar estas actividades y sus consecuencias.

Definición de geo-simulación

La geo-simulación se distingue por la combinación de múltiples características en un marco unificado, dando como resultado sistemas que suelen ser complejos, adaptables y dinámicos, centrados en la representación de las unidades elementales que componen el sistema y en las interacciones que tienen lugar[15].

Las principales características son:

- Gestión de entidades espaciales
- Gestión de relaciones espaciales: interacciones entre componentes elementales que afectan niveles superiores (de abajo hacia arriba).
- Gestión del tiempo: el manejo multiescalar y discreto del tiempo permite introducir fenómenos asincrónicos.
- Paso directo de la abstracción al mundo real y viceversa, esto permite probar hipótesis, hacer descripciones realistas y apoyarse en acontecimientos claves de la geografía.
- Combinación de métodos cualitativos y cuantitativos utilizando entornos SIG.
- Utiliza características o estados nominales, ordinales, y continuos.
- Puede emplear normas de transición deterministas, estocásticas y difusas.

Autómatas celulares

Los autómatas celulares (AC) son modelos matemáticos utilizados para simular sistemas o procesos complejos. En varios campos, incluyendo la biología, la física y la química se emplean para analizar fenómenos tales como el crecimiento de las plantas, la evolución del ADN y la embriogénesis. En los años cuarenta, John von Neumann formalizó la idea de los AC para crear un modelo teórico para una máquina auto-reproductora. El trabajo de Von Neumann fue motivado por su intento de entender la evolución biológica y la auto-reproducción [16].

Elementos de un autómata celular:

Un autómata celular es una tupla $C = (D, Q, V, \delta)$ la cual está constituida por los siguientes elementos[17]:

- **Un espacio regular (D):** ya sea una línea, un plano de 2 dimensiones o un espacio n-dimensional. Cada división homogénea del espacio es llamada célula. ($d \geq 1$ es la dimensión del autómata $r \geq 0$ es su índice de localidad).
- **Conjunto de Estados (Q):** es finito y cada elemento o célula del arreglo toma un valor de este conjunto de estados. También se denomina alfabeto. Puede ser expresado en valores o colores. (Q es el conjunto de estados) $\# \subset Q$ es un estado especial llamado quiescente. Por definición $\delta(\#, \#, \#, ...\#) = \#$. Este estado implica ausencia de actividad.
- **Vecindades (V):** define el conjunto contiguo de células y posición relativa respecto a cada una de ellas. A cada vecindad diferente corresponde un elemento del conjunto de estados. ($V = (z_1, z_2, ..., z_1) \subset (z_d)^r$ es un vector de vecindad, que contiene r elementos distintos de z_d).
- **Función Local (δ):** es la regla de evolución que determina el comportamiento del AC. Se conforma de una célula central y sus vecindades. Define como debe cambiar de estado cada célula dependiendo de los estados anteriores de sus vecindades. Puede ser una expresión algebraica o un grupo de ecuaciones. $(\delta : Q^(r + 1)) \rightarrow Q$ es la regla de transición del autómata.

Grafo

Un grafo, G, es un par ordenado de V y A, donde V es el conjunto de vértices o nodos del grafo y A es un conjunto de pares de vértices, a estos también se les llama arcos o ejes del grafo. Un vértice puede tener 0 o más aristas, pero toda arista debe unir exactamente a dos vértices.

Utilizaremos la notación $G = (V, A)$ para designar al grafo cuyos conjuntos de vértices y aristas son, respectivamente, V y A [18].

Grafo no dirigido

Un grafo no dirigido G es un par (V, A), en el que V es un conjunto cuyos elementos se denominan vértices, y A una familia de pares no ordenados de vértices, que representaran las aristas [19].

1.2. Panorama actual de aplicación SIG

Actualmente existe una gran diversidad de software SIG, cada uno de ellos con numerosas alternativas, pudiendo resultar complejo elegir la adecuada a cada necesidad; para esto es necesario tener una visión

global de sus representantes y de las características que los diferencian. A continuación, se realiza un breve análisis de algunas de las principales aplicaciones SIG. Para ello se considera la característica más destacable del software libre para SIG: su modularidad;[20] 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; aún 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 [21].

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[22].

Software Qgis

Qgis es una aplicación SIG con grandes potencialidades para la edición de mapas, multiplataforma y desarrollado utilizando Qt Toolkit[1] y C++. Ofrece muchas características SIG, entre las que se encuentran

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

[23]:

- 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.
- Incorpora a través de las herramientas de procesado, decenas de comandos de GRASS y SAGA para realizar análisis espacial tanto con datos vectoriales[2] como ráster[3].
- 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 considera 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, dando paso a la reutilización de algunas de sus funcionalidades e integración de la solución .

1.2.1. Análisis de soluciones existentes con soporte para la geo-simulación

En el presente epígrafe se realiza un estudio de las principales herramientas existentes que soportan la geo-simulación. Se considera para el análisis los tipos de datos usados para la representación del espacio celular,el campo de acción, los tipos de indicadores que analizan y el tipo de vecindad usada.

Modelo de crecimiento urbano de la ciudad de Shanghai, China (2009)

Recientemente, se desarrollo un modelo de crecimiento urbano de la ciudad de Shanghai utilizando autómatas celulares. En el modelo se incluyo el análisis de fuerzas socioeconómicas, procesos de migración poblacional, y un modelo espacial urbano [24]. En el estudio se busca identificar el cambio de la tierra urbana por variables socioeconómicas y simular el proceso de expansión urbana por factores físicos.

[2] El modelo vectorial es una estructura de datos utilizada para almacenar datos geográficos

[3] Los datos raster son una abstración de la realidad, representan ésta como una rejilla de celdas o píxeles

Desventajas:

- Utiliza datos ráster en la representación del espacio celular.
- Tiene definido un conjunto estático de datos lo que restringe el campo de análisis.
- Simula el crecimiento urbano poblacional.

Simulación del crecimiento urbano y modelos basados en autómatas celulares: el uso de parcelas catastrales vectoriales a partir de la teoría de grafos

El desarrollo de un prototipo de modelo basado en AC para la simulación del crecimiento urbano que emplea como unidad espacial básica la parcela catastral. Para poder reducir el tiempo computacional que conlleva el uso de información vectorial, se ha realizado una abstracción de la estructura espacial a partir de la teoría de grafos. El prototipo sigue el esquema NASZ (Neighbourhood, Accessibility, Suitability, Zoning) propuesto por White y Engelen (1993), donde intervienen para la determinación de los nuevos espacios urbanos factores como la vecindad, accesibilidad, aptitud y clasificación del suelo. La vecindad en este caso no es estática, de forma que varía en función del tamaño de la parcela estudiada [25].

Desventajas:

- Tiene definido un conjunto estático de datos lo que restringe el campo de análisis.
- Simula el crecimiento urbano poblacional.
- El tratamiento de algunos de los factores del modelo han de seguir siendo investigados y mejorados.

Propagación de patrones urbanos – El modelo Sleuth (2002)

Clarke y sus colegas (2002) construyeron un modelo AC llamado Sleuth (pendiente, cobertura de la tierra, exclusión, urbanismo, transporte y la sombra de la colina) [15].

El modelo realiza los cambios en la forma urbana en cuatro pasos que describen las diferentes formas de propagación y difusión en el espacio: a) urbanización espontánea, b) generación de nuevos centros de difusión, c) difusión en los bordes de las áreas urbanizadas, y d) influencia de la carretera en la difusión.

El modelo Sleuth tiene una tasa de crecimiento urbano empleada en cada una de las cuatro etapas de Sleuth. Cuando la tasa de crecimiento del grupo supera un determinado umbral, la probabilidad de urbanización espontánea y la difusión de todos los pasos se aceleran al multiplicar las tasas por un factor superior a uno.

Para evitar crecimientos explosivos, la aceleración se controla, y el multiplicador se reduce linealmente con el envejecimiento de un clúster.

Desventajas:

- Tiene definido un conjunto estático de datos lo que restringe el campo de análisis.
- Simula el crecimiento urbano poblacional.
- Utiliza datos ráster en la representación del espacio celular.

A partir del análisis realizado, teniendo en cuanta lo criterios de comparación, la Tabla 1.3 muestra los resultados obtenidos sobre las herramientas seleccionadas que dan soporte para la geo-simulación.

Tabla 1.3: *Tabla comparativa de las herramientas para el análisis prospectivo.*

Fuente: *(Elaboración propia)*

Titulo	Tipo de datos del espacio celular	Campo de acción	Vecindad Estándar	Tipo de indicadores que utiliza
Propagación de patrones urbanos – El modelo Sleuth (2002)	Ráster	Crecimiento urbano	Monroe	Estadísticos
Modelo de crecimiento urbano de la ciudad de Shanghai, China (2009)	Ráster	Crecimiento urbano	Monroe	Estadísticos
Simulación del crecimiento urbano y modelos basados en autómatas celulares: el uso de parcelas catastrales vectoriales a partir de la teoría de grafos	Vectorial	Crecimiento urbano	Dinámico	Estadísticos

1.3. Lenguajes, herramientas y tecnologías a utilizar

En todo proceso investigativo es necesario la utilización de sistemas de soporte que permitan 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 la estipuladas por el grupo de investigación: Desarrollo de componentes para el análisis espacial en salud pública, al cual está relacionada la presente investigación.

1.3.1. Lenguaje de modelado

Lenguaje Unificado de Modelado (UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Además, ofrece un estándar para describir un "plano"del sistema

(modelo), incluyendo aspectos conceptuales tales como procesos de negocios, y funciones del sistema. Sirve para hacer modelos y especificar la estructura y/o comportamiento de los mismos, hacer plantillas que guíen su construcción y documentar las decisiones que se han tomado [26].

Algunos de los beneficios que presenta UML son[26]:

- Mejores tiempos totales de desarrollo.
- Mejor soporte a la planeación y al control de proyectos.
- Mayor independencia del personal de desarrollo.
- Mayor soporte al cambio organizacional, comercial y tecnológico.

1.3.2. Herramienta CASE

Las herramientas *Computer Aided Software Engineering* (CASE) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero, se utiliza para el análisis de requisitos, el modelado de sistemas, la depuración y las pruebas, además estas herramientas pueden ayudar a todos los aspectos del ciclo de vida de progreso de un software [27].

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 [28].

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.

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 [29].

Es importante tener en cuenta que al seleccionar Qgis como el software que soporte la integración de la solución, el lenguaje de programación conveniente a utilizar es Python, que junto con el módulo PyQT entrega una solución al desarrollo de plugins e interfaces gráficas de usuario.

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 [30].

Con la utilización de 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 [31].

En correspondencia con la elección anterior de PyQt, se ha decidido emplear Qt Designer en su versión 4.7.4 como elemento que soporte 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.

1.3.3. 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[32].

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 [33].

1.3.4. 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 bases de datos.

Estos softwares facilitan el proceso de definir, construir y manipular bases de datos para diversas aplicaciones [34].

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 [35].

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 [36].

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 [37].

1.4. Metodología XP

Una metodología de desarrollo de software es “un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software”. Una metodología es la que define Quién debe hacer, Qué, Cuándo y Cómo.[38]

Existen dos grandes grupos o corrientes que agrupan las metodologías de desarrollo del software, las metodologías tradicionales o pesadas y las metodologías ágiles o ligeras. “Las primeras están pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y al mantener una buena relación con el cliente” [39].

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 la retroalimentación 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[40].

Características de la metodología XP[41]

- XP es una metodología “liviana” que no tiene en cuenta la utilización de elaborados 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[41]

- 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

La construcción del marco teórico referencial de la investigación relacionado con los modelos de geo-simulación, permitió identificar los métodos para realizar análisis prospectivo y la selección de los modelos de simulación basados en autómatas celulares que permiten construir futuros alternativos inciertos y dinámicos, desde una visión holística, utilizando técnicas exploratorias con uso normativo. La revisión del panorama actual de los SIG condujo a la selección de QGIS como el adecuado para la integración

de la solución. Durante el desarrollo del capítulo se realizó un estudio de varios sistemas existentes que soportan procesos de geo-simulación, lo que facilitó obtener una visión de la propuesta de solución. Para la implementación del software fue seleccionado un conjunto de herramientas y tecnologías basadas en licencias de software libre, siendo estas las estipuladas por el proyecto de investigación: Desarrollo de componentes para el análisis espacial en salud pública.

Capítulo 2

COMPONENTE INFORMÁTICO PARA LA GEO-SIMULACIÓN BASADA EN AUTÓMATAS CELULARES SOBRE EL SISTEMA DE INFORMACIÓN GEOGRÁFICA QGIS

En este capítulo se describen los elementos relacionados con el desarrollo de un componente informático para la geo-simulación basada en autómatas celulares sobre el Sistema de Información Geográfica QGis. 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.

2.1. Propuesta de geo-simulación basada en autómatas celulares

El proceso de geo-simulación basada en autómatas celulares está constituido por las siguientes fases:

- **Selección:** se seleccionan las características cartográficas. Se obtienen los indicadores estadísticos desde una hoja de cálculo.
- **Pre-procesamiento:** se obtiene el aporte informacional, se normalizan los datos de los indicadores seleccionados y se obtiene el aporte de riesgo de salud de cada territorio.
- **Calibración:** se configuran los elementos para la realización de la regla de evolución.
- **Simulación:** se representa la evolución de los estados de un territorio respecto al riesgo teniendo en cuenta la función local definida.

- **Visualización:** se representa en un mapa temático cada territorio.

Como parte de la propuesta de solución de la presente investigación se muestra en la Figura 2.1 un modelo conceptual donde se describen las relaciones de los componentes mediante una secuencia de pasos, que le permitirá al especialista obtener información prospectiva en cuanto a los procesos de geo-simulación basada en autómatas celulares. A partir de los elementos del modelo y sus características se identificaron los requisitos que se presentan en el epígrafe siguiente.

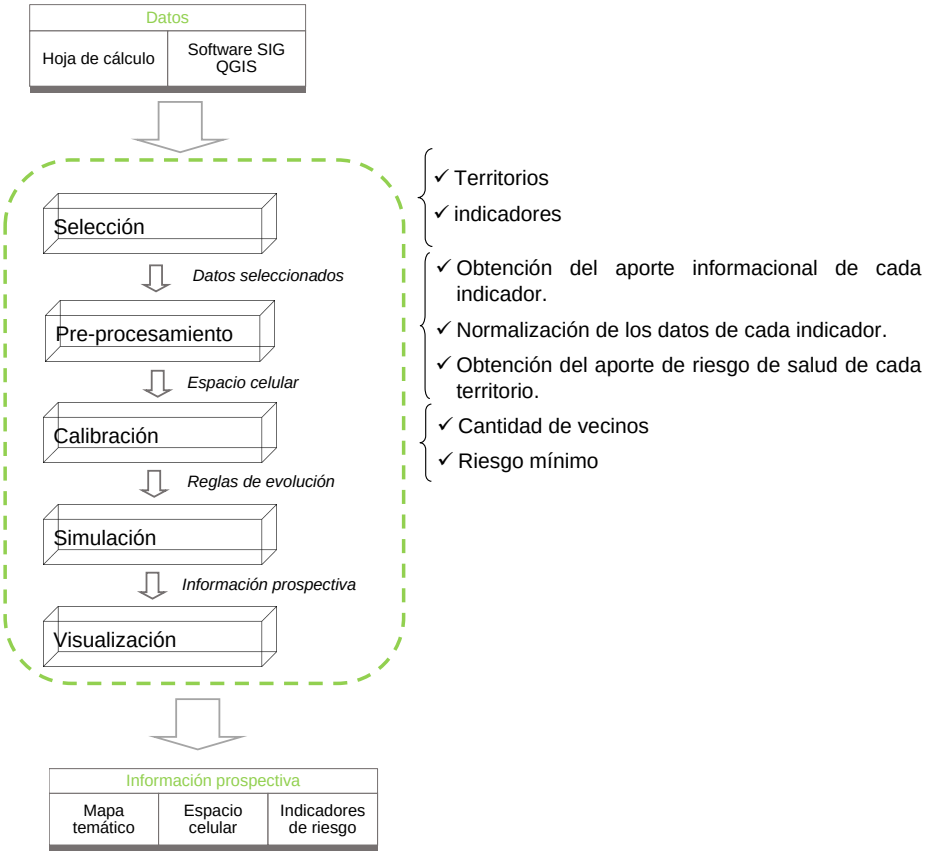

Figura 2.1: *Modelo de la propuesta de solución.*

Fuente: *(Elaboración propia)*

Como premisa para realizar el proceso de geo-simulación se debe contar con los indicadores estadísticos definidos previamente por el usuario y el empleo de la cartografía asociada a los territorios.

2.1.1. Pre-procesamiento de los datos

Obtención del aporte informacional de los indicadores

El aporte informacional de los indicadores se obtiene mediante un método estadístico que calcula el coeficiente de variación de cada uno. Debido a los diferentes dominios en los que se presentan los datos de los indicadores, se utiliza el coeficiente de variación [42].

Varianza de los datos

$$S^2 = \frac{1}{t}\sum_{i=1}^{t}(V_i - \bar{x})^2$$

Donde:

- t: Total de la población finita de los datos.
- $\bar{x}$: Promedio de los datos.
- V_i: Valor i, donde $t = 1, 2, ..., t$.

Posteriormente se obtiene la desviación estándar (S), la cual se obtiene de la raíz cuadrada de la varianza de los datos. El coeficiente de variación obtiene la dispersión de los datos en función de su promedio.
El coeficiente de variación se determina de la siguiente manera:

$$W_k = \frac{\text{desviación estándar}}{\text{media}} = \frac{S}{\bar{x}}$$

Por lo tanto, W_k es la aportación informacional del indicador k , donde $k = 1, 2, ...$, total de indicadores.

Normalización de los datos seleccionados

Para evitar que un indicador no domine sobre otro, sus valores se normalizaron mediante la siguiente función [43].

$$\frac{X_{if} - Min_f}{Max_f - Min_f}$$

Donde:

- X_{if}: Valor i del indicador f, donde $i = 1, 2, ...$, total de territorios.

- Min_f: Mínimo valor del indicador f.
- Max_f: Máximo valor del indicador f.
- $f = 1, 2, ...$, total de indicadores.

Obtención del riesgo de salud por territorio

Debido a que los territorios generados no tendrían razón alguna para el experto si no sabe cuál de estos presenta mayor riesgo de salud, se propone desarrollar una funcionalidad que etiquete los grupos obtenidos por riesgo, dividido en dos casos de evaluación del aporte de riesgo para los indicadores, debido a que en algunos indicadores un valor alto no significa que tenga mayor riesgo y viceversa.

Caso#1: A mayor valor en el dato implica mayor aportación de riesgo para el indicador[44].

$$Ap = \frac{X_{ik}}{Max_k}$$

Donde:

- Ap: Aportación de riesgo para el caso 1.
- X_{ik}: Es el valor que tiene el territorio x_i, $i = 1, ..., h$, donde h es el total de territorios, en el indicador k, $k = 1, ..., n$, donde n es el total de indicadores.
- Max_k: Es el mayor valor que toma el indicador k.

Caso#2: A mayor valor en el dato implica menor aportación de riesgo para el indicador[44].

$$Ap = 1 - \frac{X_{ik}}{Max_k}$$

Donde:

- Ap: Aportación de riesgo para el caso 2.
- X_{ik}: Es el valor que tiene el territorio x_i, $i = 1, ..., h$, donde h es el total de territorios, en el indicador k, $k = 1, ..., n$, donde n es el total de indicadores.
- Max_k: Es el mayor valor que toma el indicador k.

Finalmente se aplica una función de integración de los indicadores para obtener la aportación de riesgo final por territorio.

Función total de aportaciones de riesgo[44].

$$\Gamma_{\tau o}(X_m) = \frac{\sum_{i=1}^{nm} W_i Ap_i}{\sum_{i=1}^{nm} W_i}$$

Donde:

- $\Gamma_{\tau o}$: Es la aportación de riesgo por territorio.
- X_m: Territorio m , donde $m = 1, 2...$, total de territorios.
- Ap_i: Aportación de riesgo del indicador i.
- W_i: Aportación informacional del indicador i.
- nm: Total de indicadores.
- $i = 1, 2, ..., nm$.

2.1.2. Elementos del autómata celular

Para la creación de los elementos del autómata celular se propone utilizar los procesos que se describen a continuación.

- **Representación del espacio:** el formato raster utilizado en los modelos basados en AC tradicionales puede ser considerado como un buen método de representación cuando se estudian sistemas complejos con el ánimo de comprender su comportamiento global. El problema surge cuando se pretende simular una realidad a una escala de mayor detalle. Esto no siempre es posible resolverlo mediante la utilización de celdas de menor tamaño, pues en este contexto lo que resulta poco realista es representar de forma regular elementos que son, por naturaleza, irregulares. La complejidad de utilizar información en formato vectorial es el coste computacional necesario para leer cada uno de los territorios y realizar los análisis espaciales necesarios. Si el grado de detalle o el número total de territorios a procesar no es excesivamente grande, podría ejecutarse el AC a través del constructor de modelos en cualquier software SIG utilizando un determinado lenguaje de programación. Por el contrario, si el número de territorios es elevado o existe un alto grado de detalle que impide una rápida ejecución del modelo, surge la necesidad de buscar otras vías para hacer viable computacionalmente su ejecución. En este sentido, la teoría de grafos puede resolver este problema, haciendo eficiente el manejo de una información real de partida, de forma irregular

y en formato vectorial. Así, la solución sería realizar una abstracción de la representación vectorial de los territorios a un formato de tipo grafo [45], mediante el Algoritmo 1 se obtiene el espacio celular.

Algoritmo 1 Construir espacio celular

Entrada: Lista de territorios *Lt*.

Salida: Grafo *G*

```
1: Crear grafo G
2: para todo T_i ⊂ Lt hacer
3:    Agregar T en G
4:    para todo T_i ⊂ vecinos(T) hacer
5:       relacionar (T_i, T, G)
6:    fin para
7: fin para
```

- **La vecindad:** es un factor crítico en un modelo AC vectorial, puesto que cada territorio es irregular y a su vez distinto del resto. Por lo tanto, no puede tratarse como un factor estático (así tratado en los modelos raster). Este factor define la influencia que ejercen los territorios considerados como vecinos sobre el territorio de estudio y su efecto dependerá de la distancia que los separa. Así, es necesario identificar los territorios vecinos a cada uno de los que forman el área de estudio mediante la función ***intersects***. Esta determina si dos geometrías se interceptan espacialmente (comparten cualquier porción del espacio).
- **Conjunto de estados:** este se asocia al nivel de aporte en riesgo que posee cada territorio, se encuentra en el intervalo de 0 a 1 dado por la normalización realizada en los valores, además este se representa de forma visual mediante una paleta de colores conformada por azul-rojo donde el menor valor de riesgo sera de color azul y el mayor de color rojo.
- **Función local:** la evolución del estado de un territorio dependerá del nivel de riesgo en que se encuentren sus vecinos. Se propone el Algoritmo 2 para la descripción del proceso.

Algoritmo 2 Evolucionar

Entrada: Riesgo del territorio (r), lista de vecinos del territorio *Lv*, riesgo mínimo a considerar *Rm*, cantidad de vecinos para analizar el riesgo *Cv*

Salida: Obtención del riesgo

1: **para todo** $V_i \subset Lv$ **hacer**
2: **si** $obtenerRiesgo(V_i) < Riesgo\_mnimo$ **entonces**
3: $Lista\_vecinos\_riesgo.adicionar(V)$
4: **fin si**
5: **fin para**
6: **si** $Lista\_vecinos\_riesgo \neq \emptyset$ **AND** $cantidad\_elemetos(Lista\_vecinos\_riesgo) > Cv$ **entonces**
7: $r = mayor\_riesgo(Lista\_vecinos\_riesgo)$
8: **si no**
9: $r = menor\_riesgo(Lista\_vecinos\_riesgo)$
10: **fin si**

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". 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. [27]

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 [27].

A continuación, se muestran los RF identificados:

- **RF 1:** Importar indicadores estadísticos desde una hoja de cálculo.

- **RF 2:** Obtener características cartográficas a través de Qgis.
- **RF 3:** Construir autómata celular.
- **RF 4:** Simular factores de riesgo.
- **RF 5:** Visualizar simulación.
- **RF 6:** Exportar mapa temático de una simulación como imagen.
- **RF 7:** Exportar mapa temático de una simulación hacia una hoja de cálculo.

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 [27]. A continuación, se muestran los RNF identificados:

Requisitos de Software

- **RNF 1:** Se debe tener instalada la herramienta QGIS en su versión 2.8 o superior.
- **RNF 2:** Se debe tener instalado el gestor de base de datos 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 será utilizada por cualquier usuario con conocimientos básicos sobre geografía e informática. Debe mostrar mensajes al usuario que le ayuden en llevar a cabo la tarea que se 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 2.8 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 [46].

2.3.1. Historias de usuarios

Las historias de usuario (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 [47].

Luego de obtener las principales funcionalidades del sistema, se describen diferentes HU. En la tabla 2.1 se describe la HU asociada a simular factores de riesgo; en su elaboración se estimó un tiempo para su desarrollo de 3 semanas dado su alta prioridad para el negocio, esta tarea tiene como propósito simular factores de riesgo dado indicadores estadísticos y características cartográficas

Tabla 2.1: *Historia de usuario: Simular factores de riesgo.*

Fuente: *(Elaboración propia)*

Historia de Usuario "Simular factores de riesgo"	
Número: 4	**Nombre Historia de Usuario:** Simular factores de riesgo.
Usuario: Experto	
Iteración Asignada: 2	
Prioridad de Negocio: Alta (Alta/ Media/ Baja)	**Puntos Estimados:** 3 semanas
Riesgo de Desarrollo: Alto (Alto/ Medio/ Bajo)	**Puntos Reales:** 3 semanas
Programador responsable: Miguel Autberto Jiménez Benzol, Claudia Guerra Fernández	
Descripción: Esta tarea se realizará con el fin de que la aplicación sea capaz de simular, a partir de: • los indicadores estadísticos importados por el usuario desde una hoja de cálculo. • las características cartográficas seleccionadas por el usuario a través de QGis	
Observación:	

2.3.2. Estimación de esfuerzos por historias de usuario

En el presente epígrafe se realiza la estimación del esfuerzo por HU, se debe 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. En la Tabla 2.2 se exponen la estimación de puntos por HU donde se le asigna el máximo de prioridad a las HU que responden a las funcionalidades de mayor prioridad para el negocio y para el resto un consenso en dependencia de su prioridad.

Tabla 2.2: *Estimación de esfuerzos por historias de usuario.*

Fuente: *(Elaboración propia)*

Historia de usuario	Puntos de estimación (semanas)
HU 1: Importar indicadores estadísticos desde una hoja de cálculo.	1
HU 2: Obtener características cartográficas a través de QGis.	1
HU 3: Construir autómata celular.	3
HU 4: Simular factores de riesgo.	3
HU 5: Visualizar simulación en un mapa temático.	3
HU 6: Exportar mapa temático de una simulación como imagen.	0.5
HU 7: Exportar la simulación como una hoja de cálculo.	0.5

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 2.3 se muestra cómo quedó definido el plan de iteraciones para la solución propuesta. En esta fase se definieron cuatro iteraciones para la realización del sistema: **Iteración 1**

En esta iteración se realiza la HU relacionada a modelos de datos asociados a la abstracción de los territorios a un formato de tipo grafo, vecindad, conjunto de estados y la función de evolución. Mediante la HU Construir autómata celular.

Iteración 2

En esta iteración se realizan las HU asociadas a la evolución del estado de los territorios, a partir de la función local definida mediante la HU Simular factores de riesgo; la visualización de los resultados obtenidos en la prospección de los datos a través de la HU Visualizar simulación en un mapa temático.

Iteración 3

En esta iteración se realiza la HU relacionada con el proceso para obtener los datos cartográficos y se realizan las HU relacionadas con los procesos para identificar los indicadores estadístico, tales como: Importar los indicadores estadísticos desde una hoja de cálculo, seleccionar la capa para la identificación de los territorios y el atributo para nombrarlos.

Iteración 4

En esta iteración se realizan las HU que son descritas por los procesos de Exportar mapa temático de una simulación como imagen y Exportar mapa temático de una simulación hacia una hoja de cálculo.

Tabla 2.3: *Plan de duración de las iteraciones.*

Fuente: *(Elaboración propia)*

Iteraciones	Orden de las historias de usuario a implementar	Duración de las iteraciones (semanas)
Iteración 1	Construir autómata celular.	3
Iteración 2	Simular factores de riesgo.	6
	Visualizar simulación en un mapa temático.	
Iteración 3	Importar indicadores estadísticos desde una hoja de cálculo.	2
	Obtener características cartográficas a través de QGis.	
Iteración 4	Exportar mapa temático de una simulación como imagen.	1
	Exportar simulación hacia una hoja de cálculo.	
Total		12

2.3.4. Plan de entrega

El plan de entregas establece qué HU serán agrupadas para conformar una entrega, y el orden de implementación [48]. 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 los siguientes parámetros: el tiempo de desarrollo ideal y el grado de importancia para el cliente. En la Tabla 2.4 se presenta el plan de entregas de la aplicación informática propuesta.

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

Fuente: *(Elaboración propia)*

	Final de la 1ra Iteración	Final de la 2da Iteración	Final de la 3ra Iteración	Final de la 4ta Iteración
Módulos	1ra semana de marzo	3ra semana de abril	1ta semana de mayo	2da semana de mayo
Simulación	v1.0	v1.1	v1.2	Finalizado

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 [49].

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 [50].
En las Tablas 2.5 y 2.6 se muestran las tarjetas CRC correspondientes a las clases AutomataLineal y Territorio.

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

Fuente: *(Elaboración propia)*

Clase: AtomataLineal	
Responsabilidades	**Colaboradores**
1. Crear el espacio celular. 2. Realizar una simulación del espacio.	Territorio

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

Fuente: *(Elaboración propia)*

Clase: Territorio	
Responsabilidades	**Colaboradores**
1. Calcular el aporte de riesgo de cada territorio. 2. Crear instancias de la clase Indicador.	Indicador

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 [51].

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 [51].

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

Modelo Vista Controlador:

Modelo Vista Controlador(*Model View Controller*, **MVC** por sus siglas en inglés) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos [52].

MVC, consta de tres módulos: el Modelo, la Vista y el Controlador(ver Figura 2.2). Separa la lógica de negocio de la interfaz de usuario, facilita la evolución por separado de ambos aspectos e incrementa la reutilización y la flexibilidad [53].

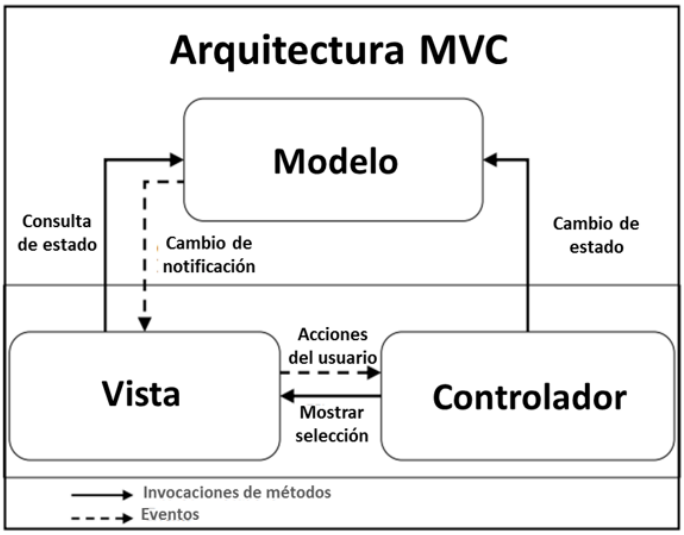

Figura 2.2: *Modelo-Vista-Controlador.*

Fuente: *(González and Romero 2012)*

En esta investigación se presenta el sistema (Sistema para el análisis espacial en salud XANGEO) perteneciente al grupo de investigación: Desarrollo de componentes para el análisis espacial en salud pública. Este está compuesto por los módulos:

- Estratificador, Servicio, Sistema para la georreferenciación y análisis de los tumores malignos (X-GATMa) y Componente para la geo-simulación basada en autómatas celulares sobre el sistema de información geográfica QGis (**GeoSimAc**) en los cuales se evidencia el patrón arquitectónico MVC como se muestra en la figura 2.3

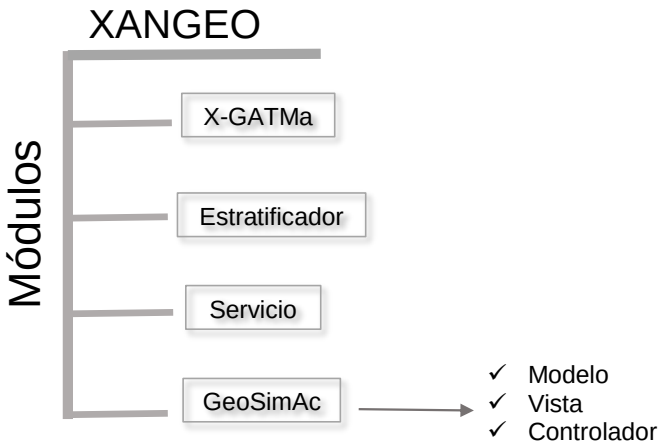

Figura 2.3: *Evidencia de la arquitectura del sistema.*

Fuente: *(Elaboración propia)*

- **El Modelo** contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- **La Vista**, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.
- **El Controlador**, actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

2.6. Diagrama Entidad-Relación

Los diagramas de entidad-relación permiten representar los elementos y los atributos, que se desean almacenar durante un tiempo determinado. Estos modelos representan la realidad a través de un esquema gráfico mediante el modelando de las entidades, atributos, relaciones, cardinalidad y llaves, que formarán la base de datos que utilizará la aplicación [62]. En la Figura 2.4 se muestra el diagrama Entidad-Relación de la aplicación informática propuesta.

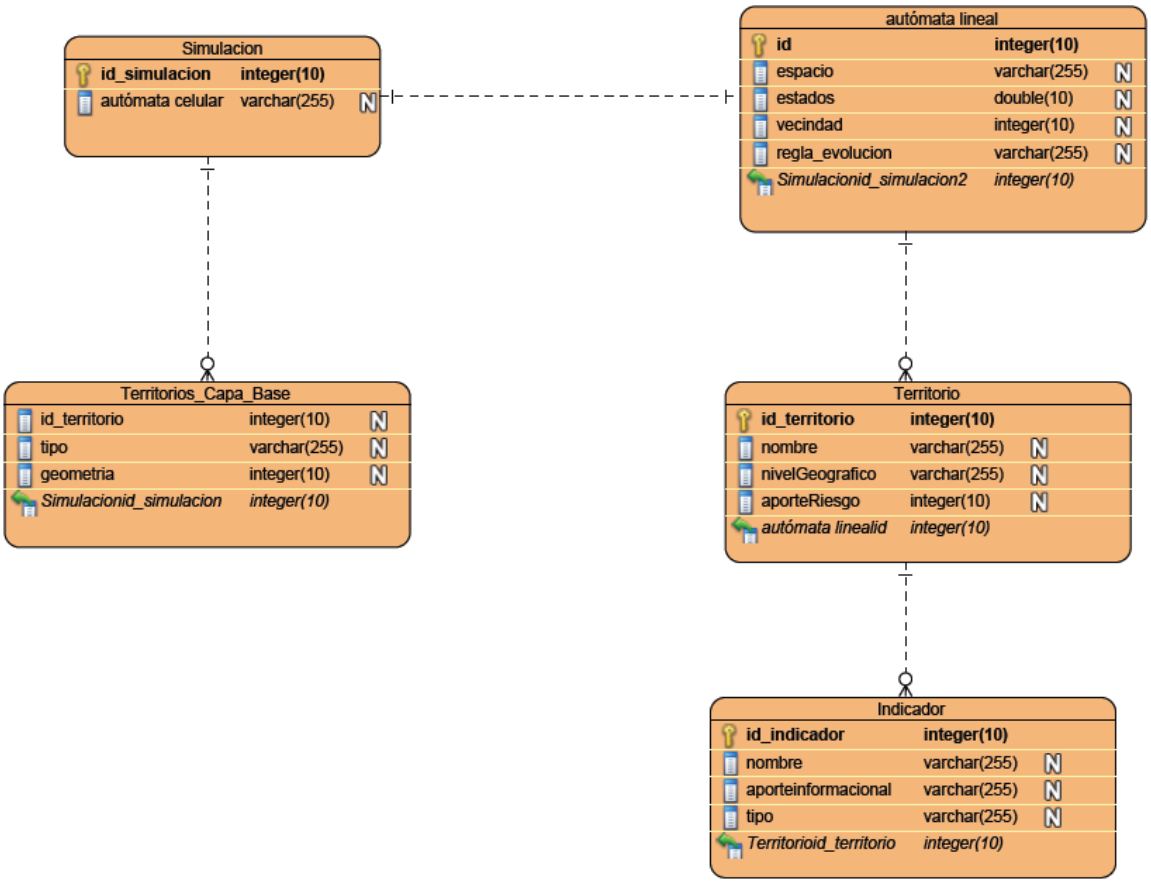

Figura 2.4: *Diagrama Entida-Relación.*

Fuente: *(Elaboración propia)*

2.7. Patrones de diseño

Un patrón de diseño es 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[55].

Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada en diferentes ocasiones, sin hacerlo ni siquiera dos veces de la misma forma [56] .

Los patrones de diseño proporcionan un esquema para refinar los subsistemas o componentes de un sistema software o las relaciones entre ellos describiendo estructuras repetitivas de comunicar componentes que

resuelven un problema de diseño en un contexto particular [57].

Entre los patrones de diseño existentes se destacan los patrones de asignación de responsabilidades GRASP (patrones generales de software para asignación de responsabilidades, siglas de G*eneral Responsability Assignment Software Patterns*) y los patrones GOF (siglas de *Gang of Four*) [55].

2.7.1. 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 a objetos [58]. Dentro de este grupo de patrones se encuentran los siguientes: experto, creador, bajo acoplamiento, alta cohesión, controlador, fabricación pura, indirección, variaciones, entre otras [57]. De ellos se utilizaron en la presente solución los siguientes:

Experto: se aplica para la asignación de responsabilidades a las clases de forma tal que las mismas contengan la información necesaria para poder ejecutar una acción específica. El uso de este patrón permitirá a los objetos valerse de su propia información para hacer lo que se les pide, favorezca la existencia de mínimas relaciones entre clases, lo que permite contar con un sistema robusto y fácil de mantener[59]. Por lo que en el presente componente informático se le asigna esta responsabilidad a la clase Territorio. En ella se evidencia dicho patrón ya que posee toda la información necesaria para calcular el aporte de riesgo del territorio. El la Figura 2.5 se muestra dicha clase.

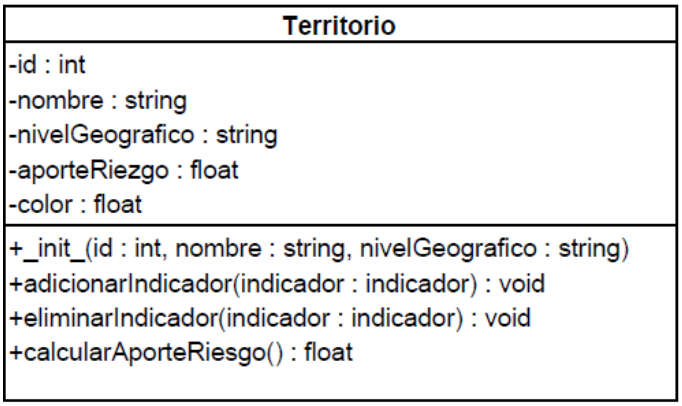

Figura 2.5: *Evidencia del patrón Experto.*

Fuente: *(Elaboración propia)*

Creador: el patrón creador ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creadora. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización[60]. En la Figura 2.6 se presenta la evidencia de dicho patrón.

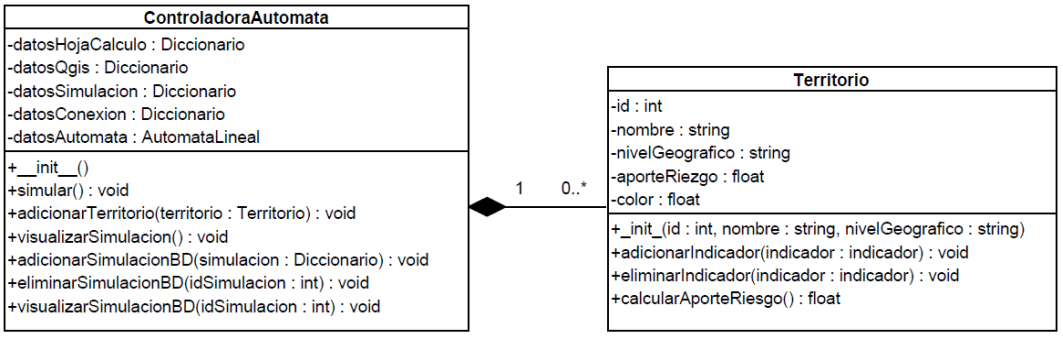

Figura 2.6: *Evidencia del patrón Creador.*

Fuente: *(Elaboración propia)*

Controlador: es un evento generado por actores externos. Se asocian con operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos. Normalmente, un controlador delega en otros objetos el trabajo que se necesita hacer, coordina o controla la actividad. No realiza mucho trabajo por sí mismo. En la Figura 2.7 se muestra dicha clase.

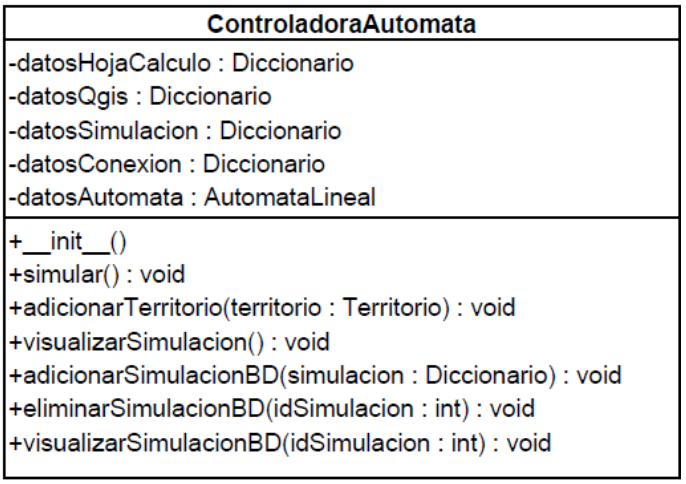

Figura 2.7: *Evidencia del patrón Controlador.*

Fuente: *(Elaboración propia)*

Alta cohesión: se aplica para realizar un diseño que evite contener clases con un alto grado de abstracción, que asuman responsabilidades que podían haber delegado a otros objetos o que tengan responsabilidades muy complejas. Se tienen las clases controladoras que se encargan de ejecutar acciones de acuerdo a las peticiones que le llegan y las clases de acceso a datos que interactúan con el modelo, de forma tal que se elimina la sobrecarga de funcionalidades en las clases controladora.

Bajo acoplamiento: este patrón apoya al diseño de clases más independientes, que no se afectan por cambios de otros componentes, potenciando la reutilización. En la aplicación se garantiza el uso de este patrón 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.

2.7.2. Patrones de GoF

Los patrones de diseño *Gang of Four* **GOF** resuelven problemas específicos de diseño de software [61]. Estos están clasificados según su ámbito, en objeto y de clase, y según su propósito en creacionales, estructurales y de comportamiento [55], estos se explican a continuación:

- **Creacionales:** los patrones creacionales abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados. Ejemplos de patrones de creación son: Abstract Factory y Factory Method [51].
- **Estructurales:** los patrones estructurales se ocupan de como las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades.Un ejemplo de patrón estructural es: Adapter[51].
- **Comportamiento:** los patrones de comportamiento están relacionados con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar,manejar y combinar comportamientos. Ejemplo de comportamiento: Chain of Responsability [51].

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 la clase AutomataLineal, esta hereda todas las funcionalidades de la clase AutomataCelular y redefine el método run() en función de sus características. En la Figura 2.8 se muestra cómo se evidencia el patrón plantilla en la aplicación informática propuesta.

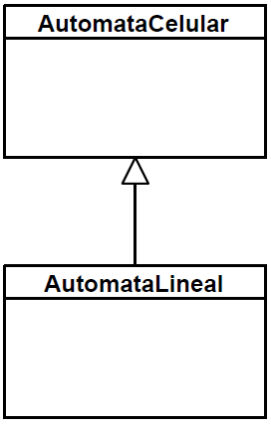

Figura 2.8: *Evidencia del patrón Método plantilla.*

Fuente: *(Elaboración propia)*

2.8. Conclusiones del capítulo

La propuesta de solución definida facilitará la realización de la geo-simulación usando como modelo matemático los autómatas celulares para el análisis de la información prospectiva en cuanto a la propagación del riesgo de los territorios basado en 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. El uso del estilo arquitectónico MVC y de los patrones de diseño GRASP y GoF permitió una mejor estructuración de la aplicación.

Capítulo 3

RESULTADOS Y VALIDACIÓN DE LA PROPUESTAS

En el capítulo se ejecutarán las pruebas y validaciones del algoritmo las cuales constituyen un instrumento para comprobar el nivel de calidad de un producto. La metodología divide las pruebas en dos grupos: pruebas unitarias, encargadas de verificar el código, la cual es realizada por los programadores, como también plantea las pruebas de aceptación destinadas a evaluar si al terminar una iteración se consiguió la funcionalidad requerida diseñadas por el cliente. Durante las iteraciones las historias de usuarios seleccionadas serán traducidas a pruebas de aceptación. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una historia de usuario ha sido implementada correctamente. Se establece el estándar de codificación que se utilizará en el desarrollo de la solución.

3.1. Fase de implementación

Una vez definidos 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 donde 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.

3.1.1. Tareas de ingeniería

Cada HU está compuesta por una o varias tareas de ingeniería, éstas se realizan para especificar las acciones llevadas a cabo por los programadores. En la Tabla 3.1 se detallan para la iteración número uno, las tareas a desarrollar por cada HU y en la Tabla 3.2 se describe una tarea de ingeniería que responde la HU construir

autómata celular, el resto se encuentran especificadas en anexos.

Tabla 3.1: *Distribución de tareas de ingeniería por HU (iteración 1).*

Fuente: *(Elaboración propia)*

HU	**Tareas de ingeniería**
Construir autómata celular.	• Normalizar los datos de los indicadores. • Obtener el aporte informacional de los indicadores. • Obtener el aporte de riesgo de salud de los territorios. • Construir el espacio celular del autómata celular • Construir la función de transición

Tabla 3.2: *Tarea de ingeniería Construir el espacio celular del autómata celular.*

Fuente: *(Elaboración propia)*

Tarea de ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** HU # 4
Nombre Tarea: Construir el espacio celular del autómata celular.	
Tipo Tarea: Desarrollo	**Puntos Estimados:** 1semana
Fecha Inicio: 13/02/2017	**Fecha Fin:** 17/02/2017
Programador Responsable: Claudia Guerra Fernández, Miguel Jimenez Benzol	
Descripción: Esta tarea permite construir el espacio celular del autómata celular mediante un grafo, donde los nodos representan los territorios y las aristas, las relaciones de vecindad entre los mismos	

3.2. 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.[41]

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.

Esto es un comentario de una línea

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 los 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.

3.3. Interfaz del sistema

A continuación se muestra en la Figura 3.1 la interfaz principal del sistema. La misma cuenta con un conjunto de funcionalidades para que los clientes puedan seleccionar, configurar y visualizar los datos asociados al proceso de simulación.

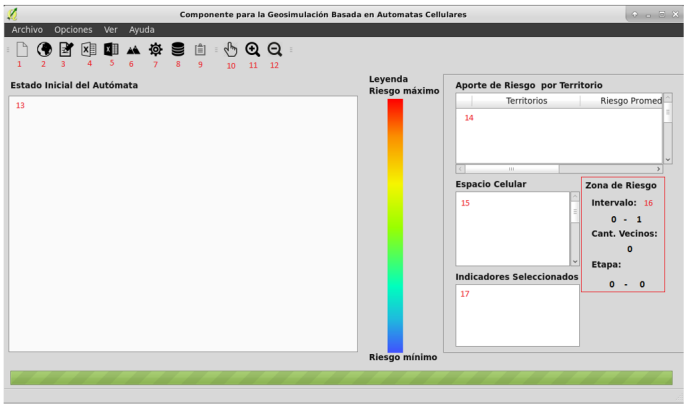

Figura 3.1: *Intrfaz del sistema.*

Fuente: *(Elaboración propia)*

- 1: Permite crear una nueva simulación.
- 2: Permite simular.
- 3: Permite cargar los datos cartográficos.
- 4: Permite cargar los indicadores estadísticos.
- 5: Permite exportar los datos de la simulación a una hoja de cálculo.
- 6: Permite exportar el mapa temático a una imagen.
- 7: Permite configurar los elementos de la simulación.
- 8: Permite importar una simulación desde una base de datos.
- 9: Permite cargar simulaciones guardadas en el historial.
- 10: Permite manipular el mapa temático.
- 11: Permite minimizar el mapa temático.
- 12: Permite maximizar el mapa temático.
- 13: Área de visualización del mapa temático.
- 14: Área de visualización de los valores asociados al aporte de riesgo por territorio.
- 15: Área de visualización del espacio celular de la simulación.
- 16: Área de visualización de los elementos que componen la función de evolución.
- 17: Área de visualización de los indicadores seleccionados en la simulación.

3.4. Pruebas

Uno de los pilares de XP es el proceso de pruebas [63]. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. Por lo que en la presente solución se realizaron pruebas para evaluar la calidad del componente desarrollado, así como para comprobar que los requerimientos de software establecidos se hayan desarrollado en su totalidad.

3.4.1. Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las Historias de Usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Las pruebas de aceptación son de gran importancia, dado que miden el grado de satisfacción del cliente con el producto desarrollado [51]. Por lo tanto, son los clientes los responsables de verificar que los datos de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, son ellos los encargados de indicar el orden de resolución de los fallos [51]. En la Tabla 3.3 se muestra el caso de prueba de aceptación aplicado a la HU Construir autómata celular.

Tabla 3.3: *Caso de prueba de aceptación Construir autómata celular.*

Fuente: *(Elaboración propia)*

Caso de prueba de aceptación	
Código: HU3_p1	**Historia de Usuario:** 3
Nombre: Construir autómata celular.	
Descripción: Prueba para validar la funcionalidad construir autómata celular.	
Condiciones de ejecución: • El sistema debe haber obtenido la capa con los territorios a evaluar, seleccionada por el usuario. • El sistema debe haber seleccionado los datos de los indicadores estadísticos • El usuario debe seleccionar la opción Configuración. • El usuario debe seleccionar al menos un territorio a evaluar. • El usuario debe seleccionar el nivel geográfico de los territorios a evaluar. • El usuario debe seleccionar al menos un indicador. • El usuario debe seleccionar un criterio de riesgo de salud para cada indicador elegido. • El usuario debe seleccionar el valor mínimo de riesgo a considerar. • El usuario debe seleccionar la cantidad de vecinos a tener en cuenta para que el territorio pueda afectar su aporte en riesgo. • El usuario debe de seleccionar la cantidad de iteraciones a realizar por la simulación. • El usuario debe seleccionar la opción Aceptar.	
Resultados esperados: En caso que se cumplan las condiciones de ejecución, el sistema realiza la simulación. En caso contrario el sistema muestra un mensaje informando el motivo por el cual no realizó la simulación.	
Evaluación de la prueba: Prueba satisfactoria	

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, 5 arrojaron el resultado esperado mientras que 3 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 exitoso. Finalmente se obtuvieron un total 8 pruebas satisfactorias de 8 casos de prueba aplicados.

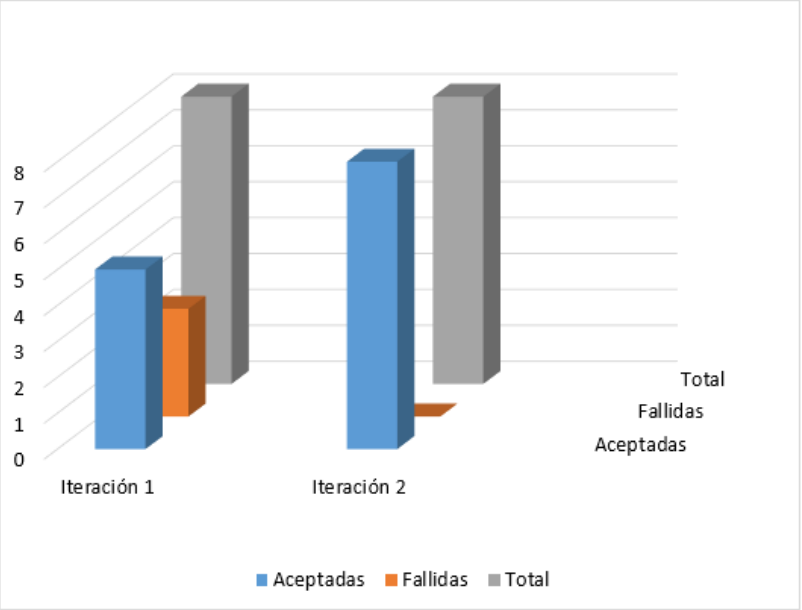

Figura 3.2: *Gráfico para la representación de los resultados de las pruebas de aceptación.*

Fuente: *(Elaboración propia)*

3.4.2. 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 [51].

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 3.3 se enumeran las sentencias de código del método crearEspacioCelular().

```
def crearEspacioCelular(self):
 (1)grafo = Grafo()
 (2)for x in self.menu.territoriosSeleccionados:
     (3)agregar(grafo, x)
     (3)gx = self.getGeometria(x.id)
     (4)for v in self.menu.territoriosSeleccionados:
         (5) gw = self.getGeometria(v.id)
         (6) if x == v:
              (7) pass
         (8) elif gx.intersects(gw) == 1:
              (9) if not Ifrelacion(grafo, x, v):
                   (10) relacionar(grafo, x, v)
(11) return grafo
```

Figura 3.3: *Código del método crearEspacioCelular().*

Fuente: *(Elaboración propia*

Luego de haberse construido el grafo se realiza el cálculo de la complejidad ciclomática[1] mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad sea correctos.

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 $V(G) = Aristas\text{-}\ Nodos + 2$.
3. La complejidad ciclomática, V(G), de un grafo de flujo G, también se define como $V(G) = Nodosdepredicado^{2} + 1$.

[1]Complejidad ciclomática: es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa.

[2]Nodo predicado: es él que representa una condicional if o case, es decir, de él salen varios caminos.

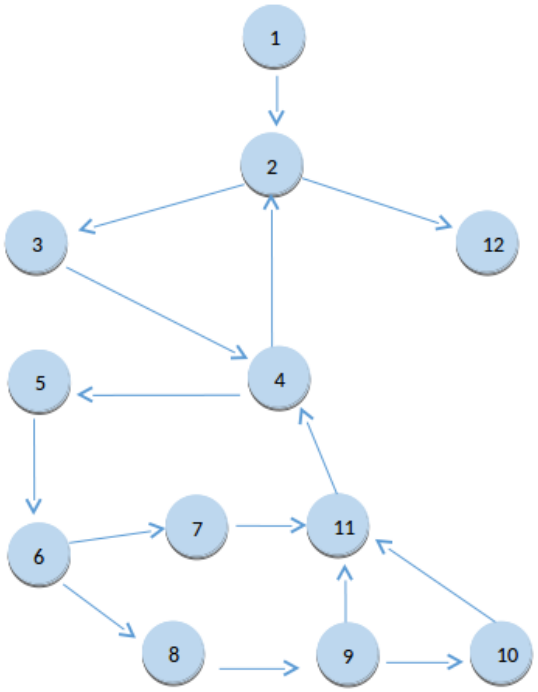

Figura 3.4: *Grafo de flujo del método crearEspacioCelular().*

Fuente: *(Elaboración propia)*

A partir del grafo de flujo del método crearEspacioCelular() que se presenta en la Figura 3.4 , la complejidad ciclomática sería:

- Como el grafo tiene cinco regiones, V(G) = 5
- Como el grafo tiene 14 aristas y 11 nodos, $V(G) = 15 - 12 + 2 = 5$
- Como el grafo tiene 4 nodos de predicado, $V(G) = 4 + 1 = 5$

Dado a que el cálculo de las tres fórmulas anteriormente mencionadas arrojó el mismo resultado por lo que la complejidad ciclomática del método es 5. Esto significa que existen 5 posibles caminos por donde el flujo puede circular. Este valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

- Camino 1: 1,2,12
- Camino 2: 1,2,3,4,2,12
- Camino 3: 1,2,3,4,5,6,7,11,4,2,12

- Camino 4: 1,2,3,4,5,6,8,9,11,4,2,12
- Camino 5: 1,2,3,4,5,6,8,9,10,11,4,2,12

Para cada camino básico determinado se realiza un diseño de caso de prueba.

Tabla 3.4: *Caso de prueba para camino básico # 1.*

Fuente: *(Elaboración propia)*

Caso de Prueba para el camino básico #1 (1,2,12)	
Descripción	Prueba para comprobar los resultados de la función crearEspacioCelular( ) en caso que la lista de territorios a evaluar sea vacía.
Condición de ejecución	• Longitud de menú.territorioselecionado = 0
Entrada	• menú.territorioselecionado = [ ]
Resultado	• grafo={ }
Resultado de la prueba	Prueba satisfactoria

Análisis de resultados

Para la validación del código generado en el desarrollo de la herramienta se seleccionaron los métodos más relevantes, a los cuales se les realizaron las pruebas para evaluar si el funcionamiento de cada uno se comportó de la manera esperada. Se realizaron un total de 12 pruebas a las 6 funcionalidades seleccionadas como relevantes, de las cuales 9 resultaron satisfactorias en una primera iteración de pruebas y 3 no satisfactorias. Estas últimas fueron solucionadas en una segunda para obtener un 100 % de pruebas satisfactorias, comprobándose la estabilidad de la lógica aplicada en el código generado en el desarrollo de la herramienta informática.

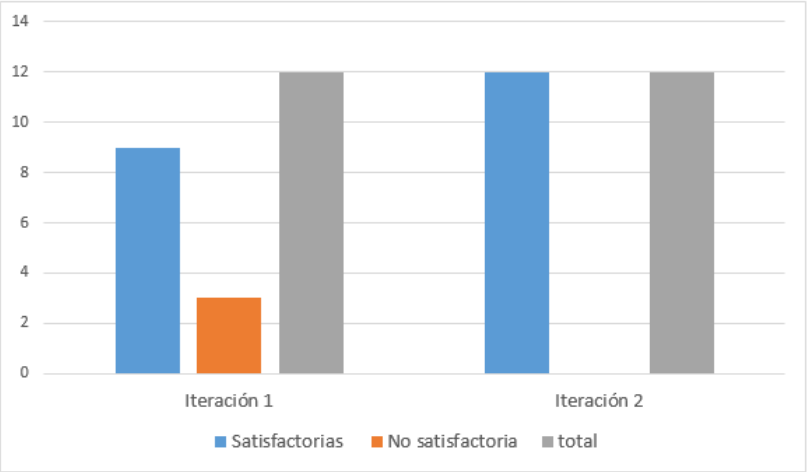

Figura 3.5: *Gráfico para la representación de los resultados de la prueba de caja blanca.*

Fuente: *(Elaboración propia)*

3.5. Caso de estudio

Se desarrolló un caso de estudio para valorar los resultados de la solución propuesta, donde se realizó un proceso de simulación de las catorce provincias de Cuba definidas en la división política-administrativa de 1976. Este estudio se aterrizó al área de la mortalidad infantil. Para su análisis se tuvo en cuenta que la mortalidad infantil ha sido definida internacionalmente como el número de defunciones que ocurren en una población pediátrica durante su primer año de vida; representa el riesgo de morir de los niños menores de un año de edad [64] por lo que se utilizó como fuente de información el Anuario Estadístico del año 2001 [7] y se seleccionaron los indicadores siguientes:

- Mortalidad infantil por cada 1000 nacidos vivos.
- Población menor de 1 año.

Aplicación del caso de estudio

Para la realización de la simulación sobre la herramienta desarrollada se determinó una zona de riesgo donde el intervalo de riesgo mínimo es de 0.35, la cantidad de vecinos necesaria para considerar una zona de peligro en 3 y la cantidad de iteraciones en 5. La Figura 3.6 muestra parte del proceso realizado.

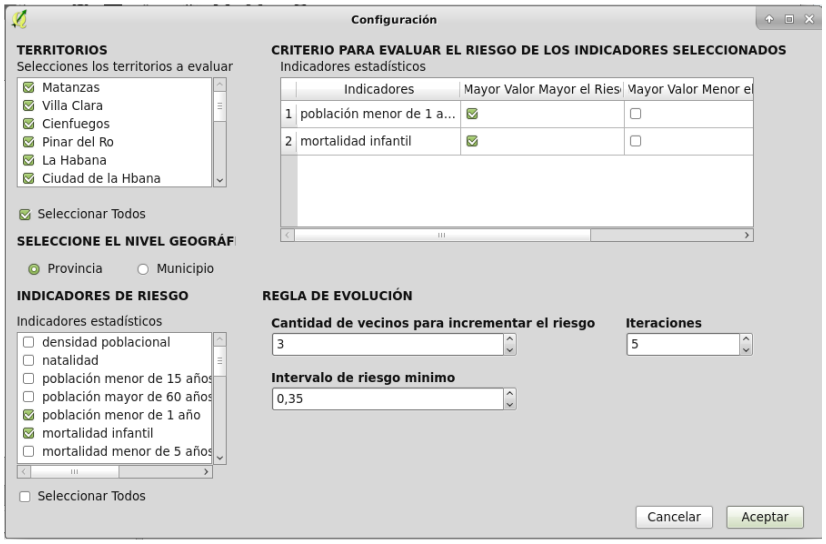

Figura 3.6: *Interfaz de usuario VistaConfiguracion.*

Fuente: *(Elaboración propia)*

Resultados de la aplicación del caso de estudio

A continuación se muestran los resultados para el estado inicial de las provincias de Cuba a partir del proceso analítico-estadístico de las variables de salud escogidas. En la Figura 3.7 se aprecian los mismos de forma mapificada y en la Tabla 3.5 de manera más detallada.

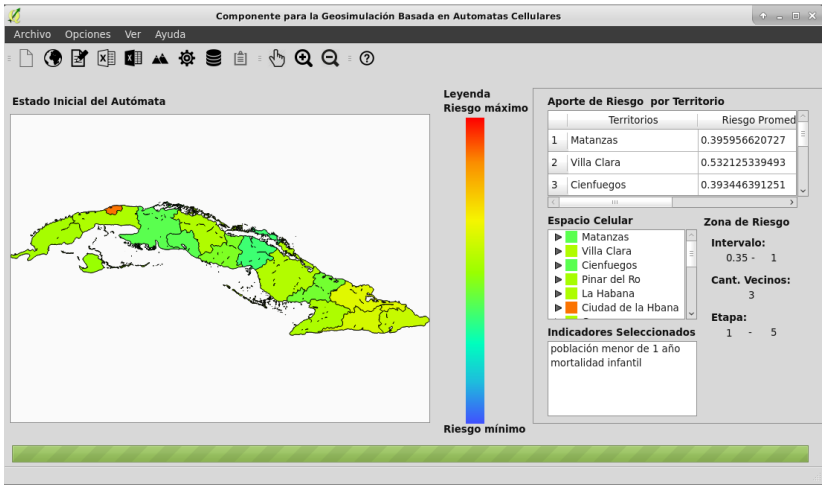

Figura 3.7: *Mapa temático del estado inicial utilizando la herramienta propuesta.*

Fuente: *(Elaboración propia)*

Tabla 3.5: *Valores del estado inicial.*

Fuente: *(Elaboración propia)*

Nombre	Aporte en riesgo
Ciego de Ávila	0.352026783
Isla de la Juventud	0.49153135
Camagüey	0.538811986
Santiago de Cuba	0.600573042
Ciudad de la Habana	0.880638622
Las Tunas	0.432353425
Guantánamo	0.573561437
Granma	0.527603643
Matanzas	0.395956621
Cienfuegos	0.393446391
Holguín	0.623657818
La Habana	0.528574582
Villa Clara	0.532125339
Pinar del Río	0.51257451
Sancti Spíritus	0.449939586

A continuación, se muestran los resultados para el estado final de las provincias de Cuba a partir del proceso de simulación analítico-estadístico de las variables de salud escogidas. En la Figura 3.8 se aprecian los mismos de forma mapificada y en la Tabla 3.6 de manera más detallada.

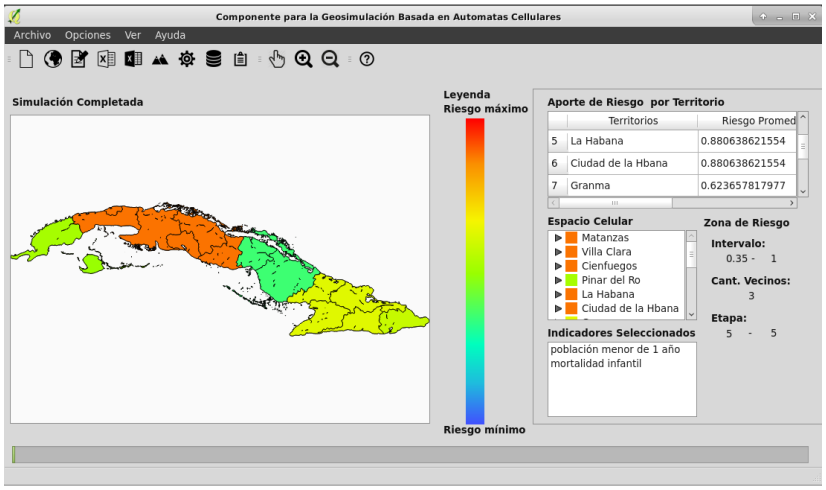

Figura 3.8: *Mapa temático de la simulación realizada utilizando la herramienta propuesta.*

Fuente: *(Elaboración propia)*

Tabla 3.6: *Resultados de la simulación realizada utilizando la herramienta propuesta.*

Fuente: *(Elaboración propia)*

Nombre	Aporte en riesgo
Ciego de Ávila	0.352026783
Isla de la Juventud	0.880638622
Camagüey	0.880638622
Santiago de Cuba	0.573561437
Ciudad de la Habana	0.880638622
Las Tunas	0.880638622
Guantánamo	0.623657818
Granma	0.880638622
Matanzas	0.352026783
Cienfuegos	0.623657818
Holguín	0.623657818
La Habana	0.880638622
Villa Clara	0.49153135
Pinar del Río	0.623657818
Sancti Spíritus	0.51257451

Las conclusiones del estudio señalan la relación del comportamiento del aporte en riesgo que presentan las catorce provincias de Cuba en cuanto a la mortalidad infantil y la población menor de un año.

3.6. Conclusiones del capítulo

En el presente capítulo se detallaron las tareas de ingeniería correspondiente a cada HU, permitiendo la organización del trabajo en una secuencia lógica de pasos. El estándar de codificación utilizado proporcionó un buen entendimiento del código y una mejor organización del mismo. 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. El caso de estudio desarrollado posibilitó realizar análisis de los resultados obtenidos en el proceso de geo-simulación basado en autómatas celulares en las provincias de Cuba.

CONCLUSIONES GENERALES

Como resultados de la presente investigación se obtuvo una propuesta de solución para la geo-simulación basada en autómatas celulares utilizando SIG. En función de los resultados obtenidos se arribó a las siguientes conclusiones:

- La definición del marco teórico referencial de la investigación relacionado con la geo-simulación basada en autómatas celulares, fundamentó la necesidad de desarrollar un plugin que se adapte a los objetivos expuestos y satisfaga las necesidades del país.
- La revisión del panorama actual de los softwares SIG permitió la selección de QGIS para la integración de la solución propuesta.
- La integración de la solución propuesta al sistema QGIS facilitó la realización del proceso de geo-simulación basada en autómatas celulares utilizando indicadores de variada naturaleza.
- Las pruebas aplicadas para la verificación de la solución informática y la valoración de los resultados a través de un caso de estudio demostró que el sistema cumple con los requisitos definidos, garantizando su correcto funcionamiento.

Recomendaciones

Los autores de este artículo recomiendan que en futuras investigaciones asociadas a esta área del conocimiento abordasen sobre:

1. Utilizar técnicas de simplificación de geometría para reducir costes computacionales del procesamiento de multipolígono.
2. Incorporar un componente para a definir nuevas reglas de transición.
3. Incorporar funciones para la evaluación de la calidad sobre la información obtenida.
4. 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.

REFERENCIAS BIBLIOGRÁFICAS

[1] G., C. J. S. G. Comparison of GIS Desktop Tools for. febrero 2013.

[2] Anaálisis espacial integral aplicado a temas de salud ambiental, 2016. Disponible en: http://www.msal.gob.ar/determinantes/images/stories/descargas/recursos/programa_analisis-espacial-integral-aplicado-a-temas-de-salud_ambiental.pdf.

[3] Gurrute Mera, N. A.; Lasso Rosero, W. R. & Vega Rivera, C. E. Establecimiento de la funcionalidad espacial de EMSSANAR para usuarios hipertensos y diabéticos utilizando herramientas SIG. 2016.

[4] Caligaris, M. G. & Rodríguez, G. B. SIMULACIONES COMPUTACIONALES: AUTÓMATAS CELULARES. agosto 2010.

[5] Alegret Rodríguez, M. *Propuestas metodológicas para la incorporación más efectiva del análisis espacial en Ciencias de la Salud*. PhD thesis, Universidad de Ciencias Médicas de Villa Clara, 2007.

[6] León, R. A. H. & González, S. C. *El proceso de investigación científica*. Editorial Universitaria, 2011.

[7] MINSAP, C. Anuario estadístico. *Dirección nacional de estadística*, 2001. Disponible en: http://bvscuba.sld.cu/anuario-estadistico-de-cuba/.

[8] Martínez, L. & M., R. *Cartografía, urbanismo y desarrollo inmobiliario*. Cie Inversiones Editoriales Dossat 2000 SL., enero 2001.

[9] Datos espaciales (SQL Server)., 2017. Disponible en: https://msdn.microsoft.com/es-es/library/bb933790.aspx.

[10] Map production., febrero 2015. Disponible en: http://icaci.org/research-agenda/map-production/Mapproduction,Mapproduction.

[11] Ruiz, C. H. & García, G. A. Propuesta de modelos predictivos en la planificación territorial y evaluación de impacto ambiental. *Scripta Nova. Revista Electrónica de Geografía y Ciencias Sociales*, 11, 2007.

[12] de Castro, A. M. G. et al. La dimensión de futuro en la construcción de la sostenibilidad institucional: proyecto nuevo paradigma. 2001.

[13] Rojas, A. F. Diseño e implementación de un modelo conceptual para la gestión integral del agua y los usos del suelo en la región de La Mojana / Design of a conceptual model its implementation for the integrated management of the water resources and land use in the region of La Mojana. Magíster en Recursos Hidráulicos, Junio 2011. Disponible en: http://www.bdigital.unal.edu.co/4144/.

[14] Porto, J. P. & Merino, M. Definición de simulación. agosto 2011. Disponible en: http://definicion.de/simulacion/.

[15] Benenson, I. & Torrens, P. M. *Geosimulation: Automata-based modeling of urban phenomena*. John Wiley & Sons, 2004.

[16] von Neumann's, J. John von Neumann's Cellular Automata. junio 2010. Disponible en: https://embryo.asu.edu/pages/john-von-neumanns-cellular-automata.

[17] Baños, A. G. VIDA ARTIFICIAL (VA) AUTODUPLICACIÓN AUTÓMATAS CELULARES EL JUEGO DE LA VIDA. 2016. Disponible en: https://campusvirtual.univalle.edu.co/moodle/pluginfile.php/774231/mod_resource/content/4/va-05_Autorreplicacion.pdf.

[18] GONZÁLEZ GUTIERREZ, F. Apuntes de Matematica Discreta. *Cádiz, España: Universidad de Cádiz*, 2004.

[19] Grimaldi, R. P. *Matemáticas discreta y combinatoria: introducción y aplicaciones*. Pearson Educación, 1998.

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

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

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

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

[24] Han, J.; Hayashi, Y.; Cao, X. & Imura, H. Application of an integrated system dynamics and cellular automata model for urban growth assessment: A case study of Shanghai, China. *Landscape and Urban Planning*, 91(3):133–141, 2009.

[25] González, P. B.; Delgado, M. G. & Benavente, F. A. Simulación del crecimiento urbano y modelos basados en autómatas celulares: el uso de parcelas catastrales vectoriales a partir de la teoría de grafos. 2015.

[26] 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.

[27] 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.

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

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

[30] What is PyQt, 2016. Disponible en: https://riverbankcomputing.com/software/pyqt/intro.

[31] Qt Documentation, 2017. Disponible en: http://doc.qt.io/.

[32] Entornos de programación. Concepto, funciones y tipos, 2012. Disponible en: http://lml.ls.fi.upm.es/ep/entornos.html#toc5.

[33] 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/.

[34] 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.

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

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

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

[38] Una revisión sistemática de la adaptación del proceso software. *Revista Española de Innovación, Calidad e Ingeniería del Software*, 3(2):39, octubre 2007.

[39] Gil, G. D.; Figueroa, A.; D., Gimson, L.; Ramírez, J. & Silvera, J. A. Metodologías ágiles y desarrollo basado en el conocimiento, evaluación cuantitativa de F/OSS para la reutilización, normas ISO y su aplicación en centros educativos. 2012.

[40] Joskowicz, J. Reglas y prácticas en eXtreme Programming. *Universidad de Vigo*, page 22, 2008.

[41] Beck, K. Extreme programming explained: embrace change. 1999.

[42] Caviedes, M. A. L. Herramienta para la estratificación de municipios en zonas de riesgo para la salud, 2004.

[43] López, J. M. M. & Herrero, J. G. *Técnicas de análisis de datos. Aplicaciones Prácticas utilizando Microsoft Excel y WEKA*. 2006.

[44] Cabrera, E. A. Nuevas extensiones del concepto de testor para diferentes tipos de funciones de semejanza, 1997.

[45] Torrens, P. M. & O'Sullivan, D. Cellular automata and urban simulation: where do we go from here?, 2001.

[46] 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.

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

[48] 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.

[49] Bustamante, D. & Rodriguez, J. Metodología Actual:Metodología XP. 2017.

[50] 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.

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

[52] Botella, A. V. CONTACTO CON MVC, 1996.

[53] González, Y. D. & Romero, Y. F. Patrón Modelo-Vista-Controlador. *Revista Telem@ tica*, 11(1):47–57, 2012.

[54] Larman, C. *UML y Patrones 2da edición*. Pearson, 2012.

[55] Gamma, E.; Helm, R.; Johnson, R. & Vlissides, J. Design Patterns: Elements of, 1995.

[56] Alexander, C. A Pattern Language: Towns, Buildings, Construction., 1977.

[57] Martínez, J. & Francisco, J. Guia de construcción de software en java con patrones de diseño, 2007.

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

[59] Alemany, F. Patrones de Diseño (GRASP), 2014. Disponible en: http://federicoalemany.com.ar/blog/ver_post/1418176832.

[60] Carmona, J. G. GRASP: Creador, 2012. Disponible en: http://juan-garcia-carmona.blogspot.com/2012/09/grasp-creador.html.

[61] 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.

[62] Rogers, P. *Ingeniería de Software un Enfoque Práctico*. 2005.

[63] Beck, K. Embracing change with extreme programming. *Computer*, 32(10):70–77, 1999.

[64] Castro Pacheco, B. L. Evolución de la mortalidad infantil en Cuba. *Revista Cubana de Pediatría*, 88(1):0–0, 2016.