Universidad de las Ciencias Informáticas

## Facultad 6

# **Título:** Herramienta computacional para el análisis de contaminación en redes de distribución hidráulica.

**Autor:** Norge Dario Concepción Pupo

**Tutores:** MSc. Gilberto Arias Naranjo.

Ing**.** Ernesto Ortega Díaz.

La Habana, Julio de 2016

"Año 58 de la Revolución"

*Prefiero equivocarme creyendo en un Dios que no existe, que equivocarme no creyendo en un Dios que existe.*

*Blaise Pascal*

# Declaración de autoría

Declaro ser autor del presente trabajo de diploma y reconozco a la Universidad de Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Norge Dario Concepción Pupo

_____

Firma del autor

MSc. Gilberto Arias Naranjo

_____

Firma del tutor

Ing. Ernesto Ortega Díaz

_____

Firma del tutor

# Agradecimientos

Agradezco al Creador del Universo por la vida, por mi familia y por los amigos que me ha dado. Le agradezco por darle sentido a mi vida. A mi esposa por ser amiga fiel, amante compañera y consejera oportuna. A mis padres por todo el amor, dedicación, preocupación y el apoyo incomparable que siempre me han dado. A mis suegros por amarme como su hijo. A todos mis familiares y amigos por el apoyo espiritual, emocional y financiero, especialmente a mis tíos Norge y Lidia, a mí cuñada Daymarelis y al Pr. Amador y familia.

Agradezco también a mis tutores, MSc Gilberto y el Ing. Ernesto Ortega, por guiarme en el desarrollo de este trabajo, por su cooperación incondicional, paciencia y acertados consejos. Agradezco a los miembros del tribunal, especialmente al Dr. Jorge Gulín González y a la Ing. Vilmavis la Rosa, por sus oportunos señalamientos y su apoyo para solucionarlos. Ellos mostraron en todo tiempo amor por la ciencia y el conocimiento.

Agradezco a cada uno de los profesores que durante estos cinco años me formaron como Ingeniero en Ciencias Informáticas. A mis compañeros de grado, especialmente a Yosbel Lázaro Guirola, Yaisel Siverio, Luis Ángel Blanco y Luis Miguel Herrera por las vivencias compartidas. A la tía Migdalia Borges por su atención en el área de extensión universitaria durante los dos últimos años de la carrera. En fin, a todos aquellos que han cooperado de una forma u otra a materializar mí sueño.

# Dedicatoria

Al Creador del Universo por ser el autor de mi existencia y de mi felicidad. A mi esposa por su amor, entrega y compañía. A mis padres por su amor, sacrificio y dedicación. A mis suegros por su amor y apoyo infinito.

# Resumen

Los sistemas de modelación de redes de distribución hidráulica automatizan el cálculo hidráulico y resuelven los requerimientos más exigentes de la calidad del agua. Sin embargo, una de las deficiencias actuales de estos programas es su incapacidad para determinar el origen de contaminación en una red de distribución. En el año 2015, como parte del proyecto de tesis "*Inferencia del origen de contaminación en redes de distribución de fluidos"* del grupo de investigación de sistemas complejos "Henri Poincaré" de la Universidad de la Habana, se desarrolló un modelo matemático que permite estimar el origen de contaminación en redes de distribución hidráulica. La validación del modelo se realiza integrando asistentes matemáticos, Epanet para el modelado de la red y rutinas aisladas implementadas en Python. Sin embargo, esto es un inconveniente para la aplicación intensiva del modelo ya que se necesita ejecutar el flujo de pasos de forma manual lo cual complica y retrasa el proceso de obtención de la estimación. Con el presente trabajo investigativo se logra automatizar este proceso haciendo uso de la biblioteca Epanet Java. Se establecen los fundamentos teóricos de la investigación y se seleccionan las herramientas adecuadas para el desarrollo de la aplicación. El proceso de construcción del software es guiado por la metodología de desarrollo OpenUP. Como resultado se obtuvo una herramienta informática que añade a la biblioteca Epanet Java funcionalidades para el análisis de contaminación y cuya eficacia para inferir el origen de contaminación en redes de distribución hidráulica se comprobó experimentalmente.

**Palabras clave:** Epanet Java, Inferencia, Origen de contaminación, Redes de distribución hidráulica, Sistemas de modelación.

# Abstract

Modeling systems of water distribution network automate the hydraulic calculation and solve the requirements most demanding of water quality. However, one of the present deficiencies of these programs is their inability to determine the source of contamination in the aqueduct. In 2015, as part of the thesis project "Inference source of contamination in fluid distribution networks" of the research group of complex systems "Henri Poincaré" of the University of Havana, a mathematical model was developed that allows estimating the source of contamination in water distribution networks (RDH). Model validation is performed integrating mathematical assistants, Epanet for modeling the network and isolated routines implemented in Python. However, this is inconvenient for intensive use of the model as it needs to run the flow of manual steps which complicates and delays the process of obtaining the estimate. In this research work is achieved to automate this process using the Epanet Java library. The theoretical foundations of the research are established and suitable for application development tools are selected. The construction process is driven software development methodology OpenUP. As result was obtained a software tool that adds functionality to the library Epanet Java for contamination analysis and whose effectiveness to infer the source of contamination in hydraulic distribution networks was experimentally tested.

**Keywords:** Epanet Java, Inference, Source of contamination, Water distribution networks, Systems modeling.

# Índice

# Índice de figuras



# Índice de tablas

# Introducción

El agua es un recurso indispensable para la vida de todos los seres vivos y para el desarrollo de cualquier nación. Aunque es un recurso renovable, es también escaso, limitado y vulnerable. Para garantizar buenas prácticas es necesario establecer políticas, que en el caso de nuestro país, están sustentadas en el uso integrado, racional y eficiente del recurso, así como en la reducción de riesgos asociados a su calidad y a situaciones extremas del clima.

El agua para abastecer a un conjunto de usuarios puede seguir diversos caminos a través de los sistemas de distribución o acueductos diseñados de una manera redundante. En estos sistemas, la difusión de contaminantes es un riesgo constante con consecuencias negativas para la salud humana y animal, por lo que evitar o eliminar en el menor tiempo posible la contaminación es de vital importancia. En los últimos años la contaminación ha aumentado y también ha decrecido la calidad de muchos depósitos de agua. La contaminación puede proceder de fuentes naturales o de la actividad humana. La más frecuente es la provocada por el hombre.

Por otra parte, el elevado número de tuberías, nodos de consumo, válvulas, depósitos de almacenamiento y equipos de bombeo que conforman las redes de abastecimiento del agua, hacen de los cálculos hidráulicos asociados un trabajo laborioso. Debido a esto se han desarrollado numerosos programas informáticos que permiten obtener los resultados hidráulicos deseados (Pardo, y otros).

Inferir el origen de contaminación en los acueductos es un problema que ha sido estudiado en los últimos años. Por ejemplo, en (Salomons, y otros, 2010) se corre el tiempo de la red hacia atrás para localizar el origen de la fuente. En (Butera, y otros, 2012) se trata el problema de la misma manera, pero se supone que la observación y el origen son conocidos y solo se trata de inferir el patrón de contaminación. En (Guan, y otros, 2006) se realiza una aproximación exhaustiva; inicialmente infieren el patrón de contaminación de cada uno de los posibles nodos orígenes y luego corren simulaciones sobre los patrones inferidos, censando los resultados en el nodo observado. Finalmente, se asume como origen de la fuente aquel que arrojó el patrón de censado más parecido al patrón observado. Estas investigaciones han sido validadas con sistemas informáticos, por ejemplo en (Salomons, y otros, 2010) se utilizó Epanet (EPA, 2008) como motor de cálculo hidráulico.

En (Ortega Díaz, 2015) se desarrolló otro modelo matemático que permite estimar el más probable origen y su patrón de contaminación en redes de distribución hidráulica. Para desarrollarlo, se realizó una simplificación de la dinámica de fluidos donde se consideran a los contaminantes de concentración binaria y a las demandas constantes. La simplificación de la dinámica de fluidos es compensada por el hecho que

se infiere el origen y el patrón de contaminación sin necesidad de mucha información, solamente el patrón de tiempos de contaminación detectado en uno o dos nodos de la red. La distribución de una sustancia o contaminante por la red es nombrada problema directo, mientras que estimar el origen de contaminación y su patrón de vertido a partir del patrón censado en uno de los nodos de la red se nombra problema inverso. Por otro lado, los acueductos son modelados como grafos conexos, direccionados y acíclicos; cuyas aristas son pesadas con el tiempo en el que el fluido demora en pasar por una tubería. Luego de varias transformaciones y simplificaciones del grafo inicial se procede a modelar el grafo final como un problema de optimización, cuyas soluciones se obtienen mediante Programación Lineal Entera.

La validación del modelo se realizó integrando asistentes matemáticos, Epanet (Rossman, 2000) para el modelado de la red y rutinas aisladas implementadas en Python. Sin embargo, esto es un inconveniente para la aplicación intensiva del modelo ya que se necesita ejecutar el flujo de pasos de forma manual lo cual complica y retrasa el proceso de obtención de la estimación.

Por lo anterior expuesto se establece como **problema a resolver**: ¿Cómo automatizar el problema directo e inverso de la propagación de contaminación en redes de distribución hidráulica mediante la dinámica simplificada?

Para darle solución al problema planteado se necesita una investigación cuyo **objeto de estudio** es: Los sistemas informáticos de modelación para redes de distribución hidráulica.

Para perfilar este estudio se define como **campo de acción**: Problema directo e inverso de la distribución de contaminación en los sistemas informáticos de modelación de redes de distribución hidráulica.

La investigación persigue como **objetivo general**: Desarrollar una herramienta informática que automatice los problemas directo e inverso de la propagación de contaminación en redes de distribución hidráulica mediante la dinámica simplificada.

Para dar cumplimiento al objetivo general se establecen las siguientes **tareas de la investigación**:

1. Revisión de la bibliografía sobre los siguientes temas:
    a. Sistemas informáticos de modelación hidráulica de redes de distribución.
    b. Procesos de la dinámica simplificada en redes de distribución hidráulica.
2. Identificación de los requerimientos de la herramienta.
3. Estudio de los sistemas informáticos para modelación de redes de distribución hidráulica.
4. Documentación de los artefactos de análisis y diseño de la herramienta.
5. Implementación de las funcionalidades de la herramienta.
6. Diseño de los casos de prueba correspondientes a las funcionalidades de la herramienta.

7. Ejecución de las pruebas a la herramienta.
8. Validación de la herramienta.

En el transcurso del estudio se hace uso de métodos y técnicas de investigación. Entre los métodos teóricos se emplea el **método histórico lógico** al revisar la evolución de los sistemas de modelación hidráulica de redes de distribución y de la trayectoria de los aportes científicos que contribuyeron a ello. El **método analítico sintético** en la realización de resúmenes y valoraciones de conceptos relevantes del objeto de estudio y del campo de acción que se estudia. Dentro de los métodos empíricos se utiliza **la observación** de los programas informáticos del área, especialmente sus funcionalidades, con el objetivo de valorar cuáles mejorar e incorporar en la solución informática que se pretender desarrollar. **La modelación** se utiliza fundamentalmente para modelar los artefactos que se generan a lo largo de todo el proceso de desarrollo de software de la herramienta.

El presente trabajo de diploma está estructurado en tres capítulos, definidos de la siguiente manera:

**Capítulo 1**. **Fundamentación teórica**: En este capítulo se abordan conceptos asociados a las redes de distribución hidráulica y se realiza un análisis de los sistemas informáticos que permiten su modelación. Además, se explica cómo se modela la distribución de una sustancia que viaja a través de la red; y cómo se realiza la inferencia del origen de contaminación, a partir del modelo desarrollado en (Ortega Díaz, 2015). Finalmente, se analizan las tecnologías y herramientas seleccionadas para darle solución al problema planteado.

**Capítulo 2**. **Análisis y diseño**: En este capítulo se realiza el análisis de los requisitos funcionales y no funcionales de la herramienta. Además, se modelan los artefactos que permiten comprender el diseño de la solución propuesta y cómo esta se integra en Epanet Java. Además, se ejemplifican los patrones de diseño que se utilizaron. Finalmente, se propone un procedimiento para la automatización del análisis de contaminación mediante el problema directo e inverso de la dinámica simplificada.

**Capítulo 3. Implementación y prueba:** En este capítulo se definen los estándares de codificación y la convención de nombres que se utiliza durante la implementación. Además, se describen los diferentes componentes de la herramienta y las pruebas aplicadas. Finalmente, se muestran los resultados de la validación.

# Capítulo 1: Fundamentación teórica

En este capítulo se muestran los principales conceptos asociados a los sistemas de modelación de sistemas de distribución hidráulica, así como una síntesis del proceso de inferencia del origen de contaminación en una red de distribución de fluidos propuesto por (Ortega Díaz, 2015). Además, se presentan las tecnologías seleccionadas para el desarrollo de la herramienta computacional.

## 1.1 Conceptos asociados al dominio del problema

Una red de distribución hidráulica es un conjunto de elementos interconectados que conducen el agua desde los puntos de alimentación a los de consumo. Los sistemas de distribución se pueden clasificar según su tipología en ramificados, mallados o mixtos, siendo estos últimos los más utilizados. Estos sistemas están compuestos por una serie de elementos cuyo comportamiento hidráulico está definido.

### *1.1.1 Elementos que componen los sistemas de distribución hidráulica*

**Tuberías:** Conducto formado de tubos por donde se lleva el agua.

**Válvulas:** Mecanismo que regula el flujo de la comunicación entre dos partes de una máquina o sistema.

**Embalses:** Gran depósito que se forma artificialmente, por lo común cerrando la boca de un valle mediante un dique o presa, y en el que se almacenan las aguas de un río o arroyo, a fin de utilizarlas en el riego de terrenos, en el abastecimiento de poblaciones, en la producción de energía eléctrica.

**Depósitos:** Lugar donde se almacena agua, pero cuya capacidad de almacenamiento es más limitada que los embalses.

### *1.1.2 Modelación para el análisis de redes de distribución hidráulica*

El Análisis Hidráulico de Sistemas de Distribución (Conocido también como Modelación Hidráulica de Redes), es en esencia la implementación de un *método numérico* asociado a redes malladas para la *resolución de un sistema de ecuaciones* de carga y de caudal. El conjunto de ecuaciones conformado por ecuaciones de carga o altura y de caudal, queda representado por un sistema de expresiones matemáticas no lineal e indeterminado que precisa la adopción de metodologías matemáticas de naturaleza iterativa que permitan finalmente determinar las distintas incógnitas del sistema.

En la práctica, la representación y análisis hidráulico de los sistemas o redes de distribución se realiza a través de un Modelo Computacional o Modelo Hidráulico que permita la solución matemática de las incógnitas del sistema de ecuaciones. Este modelo no solamente representa tuberías sino también tanques de almacenamiento, embalses (reservorios), válvulas de regulación, bombas, medidores, accesorios, entre otros elementos.

### *1.1.3 Métodos de análisis de redes de distribución hidráulica*

Los principales aportes sobre la mecánica de fluidos surgen en el siglo XVIII. En este siglo, son relevantes los aportes de Bernoulli, quien contribuyó con los principios básicos de la mecánica de fluido; y las ecuaciones de energía planteadas por Leonard Euler las cuales aún sirven de base en los modelos hidráulicos actuales.

En los primeros años del siglo XX se debe destacar la teoría de capa límite que estudiaba las reacciones que ocurren entre los fluidos y las paredes de las tuberías, así como la formulación de la ecuación *Hazen-Williams* para el cálculo de perdida de carga en las tuberías, la cual fue ampliamente adoptada en Norte América.

En 1936, Hardy Cross, un Ingeniero Estructural de la Universidad de Illinois, desarrolló un método matemático para resolver un análisis de distribución de momentos en estructuras aporticadas y que además servía para resolver caudales y presiones en redes malladas de distribución de agua.

En 1957 los investigadores Hoag y Weinberg adaptaron el método de Hardy Cross para resolver redes malladas para computadores digitales y aplicaron dicho método para el sistema de distribución de la ciudad Palo Alto en California.

Subsecuentemente, dos firmas americanas de ingeniería, Rader & Associates y Brown & Caldwell, emergieron como los pioneros en el uso del computador para analizar redes de distribución de agua para sus clientes. En el mismo año, la firma de informática Datics Corporation en Texas se convirtió en una de las primeras empresas en comercializar un programa con este fin. Así surgió la era del software para análisis de redes hisráulicas.

En la década del 60, muchos investigadores comenzaron a cuestionar el uso del método Hardy Cross para analizar caudales y presiones en sistemas de distribución debido al reconocimiento de limitaciones del método. En respuesta a ello, los investigadores comenzaron a realizar nuevos análisis con el fin de aprovechar mejor el creciente desarrollo de las computadoras.

Los métodos más destacados que se desarrollaron posteriormente fueron: el Método del Nodo Simultáneo publicado en 1963 por D.W. Martin y G. Peters, el Método del Circuito Simultaneo de Alvin Fowler y su asistente Robert Epp de la Universidad de Britosh Columbia (CA), el Método de Teoría Lineal de los profesores Don J. Wood y Charles, y el Método del Gradiente propuesto por Todini y Pilati en 1987.
Algunos de estos métodos han seguido siendo desarrollados y utilizados en sistemas informáticos comerciales. Sin embargo, el más adoptado por los sistemas de modelación de redes de distribución hidráulica, incluyendo a Epanet, es el Método del Gradiente.

## 1.2 Dinámica simplificada

La contaminación de las redes de distribución es un problema relevante que desafía a los ingenieros (García Alcaraz, 2006). Ellos conocen el grafo de la red, el tiempo que demora el agua por las tuberías, tienen nodos censados con sus respectivos patrones de contaminación y desean inferir el origen de contaminación y su patrón de vertido. En la tesis (Ortega Díaz, 2015) se presenta una solución que resuelve el problema inverso y el problema directo. A continuación se presenta una síntesis de las principales ideas planteadas en ese trabajo:

La dinámica de los fluidos no es un problema simple por lo tanto se asumen ciertas suposiciones y simplificaciones que facilitan el trabajo con las redes de distribución hidráulica:

- Se asume que el tiempo es discreto.
- Se considera la contaminación binaria (no se toma la concentración del contaminante).
- La contaminación es un evento raro en el sistema, por lo tanto lo más probable es que el origen involucre la menor cantidad de nodos en la menor cantidad de tiempos de vertido (principio de parsimonia).

El estado de los nodos del grafo que modelan la red es representado con dos índices: $i$ que representa la posición de un nodo en el grafo y $t$ que representa el tiempo.

$$S_i^t = \begin{cases} 0\ no\ hay\ contaminación\ presente \\ 1\ hay\ contaminación\ presente \end{cases}$$

El conjunto de todos los nodos padres del nodo $i$ se define como:

$$\delta^+ i = \{j: j \to i \ \in G\}$$

El conjunto de todos los nodos antecesores del nodo $i$ se define como:

$$\delta^{++} i = \{j: j \to i \ \in G\}$$

El conjunto de todos los nodos hijos del nodo $i$ se define como:

$$\delta^- i = \{j: i \to j \ \in G\}$$

El conjunto de todos los nodos sucesores del nodo $i$ se define como:

$$\delta^{--} i = \{j: i \to j \ \in G\}$$

Es decir, todo lo que se vierta en un nodo padre del nodo $i$ pasa por $i$ y llega a todos los nodos hijos de $i$.
Un patrón de vertido se define por un vector. Por ejemplo: $y_i = (1,0,1,1,0,1,0,1,1,0,1,0,)$ representa el patrón de contaminación del nodo $i$, en el cual se detectó contaminación en los instantes de tiempo 1, 3, 4, 6, 8, 9,11.

Para representar la contaminación que se vierte en el nodo $i$ se introduce un nodo $i'$ cuyo patrón de contaminación está dado por $v_i^t$. Que se puede escribir de la forma:

$v_i = (1,0,0,0,0,1,\dots,t_n)$. Siendo $t_n$ el tiempo máximo de vertido.

También se introduce una variable de tipo compuerta:

$\varepsilon_i = \begin{cases} 0\ está\ cerrada\ la\ compuerta \\ 1\ está\ abierta\ la\ compuerta \end{cases}$ que representa la conexión entre los nodos $i\ -\ i'$.

### *1.2.1 El problema directo*

Teniendo presente esto, se describe el problema directo a través de la siguiente ecuación:

$$S_i^t \equiv (\varepsilon_i\, v_i^t) \bigvee_{j \in \delta^+ i} S_j^{t-\Delta_{ji}} \qquad (1.1)$$

Esto significa que el estado del nodo $i$ en el tiempo $t$ está determinado por la contaminación que se vertió en él o bien por el estado de sus padres. El tiempo que demora el fluido del nodo $j$ al nodo $i$ es $\Delta_{ji}$. Por lo tanto, si alguno de los nodos padres del nodo $i$ estuvo contaminado en un tiempo igual a $t - \Delta_{ji}$, el nodo $i$ también lo estará en el tiempo $t$.

Para la dinámica simplificada del problema directo se creó el siguiente algoritmo (**Algoritmo 1**):

---

**Input:** Conectividad media c, número de nodos n del grafo, $\Delta_{max}$ que es el peso máximo que se le puede asignar a una tubería y $T_{max}$ duración máxima del vertido.

**Output:** $S_i^t$ de todos los nodos del sistema en todos los tiempos

---

1. Generar grafo G con n nodos.
2. Generar el patrón de contaminación $y_t$.
3. Asignar el patrón $y_t$ al nodo i de G.
4. Problema directo: distribuye contaminación en G según ecuación 1.1
5. **return** $S_i^t \forall i\ \in G$ durante todos los tiempos.

---

***Algoritmo 1 Dinámica simplificada del problema directo.*** *(Ortega Díaz, 2015)*

### *1.2.2 El problema inverso*

Para poder inferir el origen de contaminación es necesario conocer la dependencia de los estados $S_i^t$ no solo de los estados $S_j^t$ (padres) sino también el estado de todos los $\delta^{++} i$ (antecesores) y el tiempo de todos los posibles caminos hasta i.

Debido a esto es necesario redefinir la ecuación (1.1):

$$S_i^t \equiv (\varepsilon_i \, v_i^t) \bigvee_{j \in \delta^{++} i} \bigvee_{c \in C_{ij}} S_j^{t-\Delta_c^+} \qquad (1.2)$$

En este caso, el estado del nodo $i$ en el tiempo $t$ está determinado por la contaminación que se vertió en él o bien por el estado de todos sus antecesores. El nodo $j$ es un antecesor del nodo $i$ al que está unido a través del camino $c$. El tiempo que demora el fluido del nodo $j$ al nodo $i$ es $\Delta_c^+$ Por lo tanto, si alguno de los nodos antecesores del nodo $i$ estuvo contaminado en un tiempo igual a $t - \Delta_c^+$, el nodo $i$ también lo estará en el tiempo $t$.

El sistema de distribución puede ser modelado a través de un grafo $G\ (V, E)$ donde V representa el conjunto de todos los nodos y $E$ es el conjunto de todas las tuberías que conectan la red. Además cada arista de $E$ tiene un peso $\Delta_{i,j}$ que describe el tiempo que le toma a una porción de fluido recorrer ese enlace.

Cada nodo real de la red $n_i$ puede ser sustituido por un conjunto de nodos $s_i^t$ tales que cada uno representa una de las variables que describe el estado del nodo $i$ en cada instante de tiempo de la contaminación. Entonces se define el grafo $G'(V', E')$ como el grafo $G\ (V, E)$ replicado en el tiempo, en el cual las conexiones de los nodos se dan entre nodos de grafos de distintos tiempos.

De ahí la definición del grafo $G'(V', E')$ en la que $\mathrm{V}' = \left\{ s_i^t \,\middle|\, i \in V \ \wedge\ t \in [0, 1\ \ldots\ T] \right\}$

y $E' = \left\{ \left( s_j^{t-\Delta_c^+}, s_i^t \right) \middle| i\ \in V\ , j\ \in \delta^{++} i\ , c \in C_{ij} \right\}$.

El grafo $G'$ se puede reducir, pues los nodos desconectados a los nodos observados (censados) no pudieron ser origen de contaminación. De ahí la definición del grafo $G_o'$ que contiene el conjunto de nodos observados replicados en el tiempo y el conjunto de nodos conectados con los nodos observados y las aristas que los relaciona.

Si uno de los nodos observados en un instante de tiempo $t$ no está contaminado, entonces el nodo $s_i^t$ correspondiente puede ser eliminado, al igual que los nodos no observados que se conectan con él y por ende las aristas que los unen. Ahora, si un nodo observado como contaminado está en contacto con otro nodo observado y contaminado la explicación más eficiente de su contaminación es simplemente esa.

Por lo tanto, se puede definir al grafo $G_f' \left( V_f', E_f' \right)$ en el que $V_f'$ es el conjunto de nodos observados contaminados que no están en contacto con algún otro nodo observado previamente contaminado y los nodos conectados a ellos; y $E_f'$ el conjunto de aristas que los une.

El grafo $G_f'$ posee entonces dos características: es bipartido y acíclico. Esto significa que hay solo dos tipos de conjunto de nodos los observados contaminados y los conectados a ellos. Las aristas del grafo no son dirigidas, más bien indican que si uno de los nodos está contaminado todos los que estén conectado con él también lo estarán.

Los pasos necesarios para transformar el grafo $G\ (V, E)$ en $G_f'$ quedan resumidos en el siguiente algoritmo (**Algoritmo 2**):

---

**Entrada:** Grafo direccionado, acíclico y pesado $G\ (V, E)$ y conjunto de nodos observados O, y el patrón observado en ellos.

**Salida:** Grafo $G_f'$ con las variables $s_i^t$ involucradas en la explicación de la observación.

---

1. Replicar los nodos espaciales del grafo original G en nodos $s_i^t$ caracterizados por su posición en la red y el instante de tiempo $t$.
2. Generar grafo direccionado $G'(V', E')$.
3. Generar el Grafo $G_o'$ a partir del $G'$ retirando los nodos que no están en contacto con los nodos observados.
4. Generar el grafo $G_f'$ a partir de $G_o'$.
5. **return** $G_f'$

---

***Algoritmo 2 Transformación del grafo G(V,E) en G´f*** *(Ortega Díaz, 2015)*

Llegado a este punto se afirma que existe un subconjunto $X$ que pertenece a $V_f'$, capaz de explicar la contaminación detectada.

Si se define a $Y$ como un conjunto de variables binarias que indican si los de nodos observados como contaminados de $G_f'$ están o no en un conjunto capaz de explicar la contaminación detectada y a $X$ como un conjunto de variables binarias que indica si el resto de los nodos en $G_f'$ están o no en un conjunto capaz de explicar la contaminación detectada. Entonces se puede definir la siguiente ecuación lineal:

$$\forall_{y_i \epsilon\, Y}\, y_i + \sum_{x_k \in \partial y_i} x_k >= 1 \qquad (1.3)$$

Se entiende por $x_k \in \partial y_i$ al conjunto de variables binarias que pertenecen a $X$ tal que $x_k$ y $y_i$ están conectados. Esto obliga a que el nodo observado como contaminado o sus nodos vecinos tienen que haber sido origen de contaminación.

Como la contaminación se considera un evento poco común entonces la explicación más probable es la que involucre una menor cantidad de nodos del grafo $G_f'$. Lo cual puede ser descrito de la siguiente forma:

$$\min \sum_i a_i x_i + \sum_j b_j y_j \qquad (1.4)$$

Donde $a_i$ y $b_j$ son constantes que representan el peso de cada variable. Luego, este problema puede ser resuelto a través de programación lineal entera.

Para obtener el conjunto de todas las soluciones óptimas que explican este problema fue creado el siguiente algoritmo (**Algoritmo 3**):

---

**Entrada** $\boldsymbol{X_0} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \ldots, \boldsymbol{y_1}, \boldsymbol{y_2}, \ldots.\}$ solución óptima y $\boldsymbol{A_0} = \{\boldsymbol{a_1}, \boldsymbol{a_2}, \boldsymbol{a_3}, \ldots, \boldsymbol{b_1}, \boldsymbol{b_2}, \ldots.\}$ vector de los pesos de $\boldsymbol{X_0}$.
**Salida:** Conjunto de posibles orígenes de la contaminación con menor cantidad de nodos involucrados.

---

**function solucionRecursiva ($A_0$ , $X_0$)**
    $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \boldsymbol{X_0}$ // agrega $\boldsymbol{X_0}$ al conjunto de soluciones
    **for (**$\boldsymbol{x_k} \in \boldsymbol{X_0} \wedge \boldsymbol{y_q} \in \boldsymbol{X_0}$**) do**
        $\boldsymbol{a_k}$ = 1.1 ∨ $\boldsymbol{b_q}$ = 1.1 // aumenta el peso de las variables
        Obtener $X_1$ al resolver el sistema de ecuaciones lineales (1.4) con restricciones lineales (1.3) usando programación lineal.
        **If** $\boldsymbol{X_1} \in \boldsymbol{X}$ **then**
            $\boldsymbol{a_k}$ **= 1** ∨ $\boldsymbol{b_q}$ **= 1**
        **else**
            $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{\boldsymbol{X_1}\}$ // agrega la solución $X_1$ a X
            **solucionRecursiva (A1 , X1)** // paso recursivo.
            $\boldsymbol{a_k}$ **= 1** ∨ $\boldsymbol{b_q}$ **= 1**
        **end if**
    **end for**
**end function**
    S = { } // ninguna solución inicialmente
    **Solución recursiva (A0,$X_0$)**
**return S**

---

***Algoritmo 3 Multiplicidad de la solución*** *(Ortega Díaz, 2015)*

*"(…) se obtuvo que la eficiencia del método aumenta mientras mayor es el número de sensores en el sistema. También se obtuvo que la eficiencia del método disminuye cuando aumenta la conectividad media de los grafos".* (Ortega Díaz, 2015)

## 1.3 Sistemas informáticos de modelación para el análisis de redes de distribución.

Los programas informáticos diseñados para el análisis de redes de distribución producen soluciones para el diseño, construcción y manejo de la infraestructura del modelo real (García Alcaraz, 2006). Estos sistemas permiten realizar análisis hidráulico a partir de las características físicas de la red, determinando las presiones en diversos puntos del sistema, los caudales, las velocidades y pérdidas en las tuberías; así como muchos parámetros operativos derivados de los elementos presentes en el modelo real. Además, admiten extender sus potencialidades a temas de gestión como son los análisis de vulnerabilidades, análisis de protección contra incendio, estimación de costo energético, calibración hidráulica, optimización, entre otros. Asimismo, conceden la posibilidad de realizar análisis de la calidad del agua, como es la modelación del movimiento y destino de un producto desinfectante que crece o se desintegra en el tiempo a través de la red. Algunos de los más conocidos son: HidroNet[1], WaterCAD[2], WaterGEMS[2], KYpipe[3], y Epanet.

De entre todos ellos, destaca Epanet, un programa informático libre y gratuito desarrollado por la Environmental Protection Agency (EPA) capaz de realizar simulación hidráulica y de parámetros de calidad de agua en periodo extendido en redes de distribución de agua a presión. Epanet es el programa informático más utilizado en todo el mundo en el campo de la hidráulica; ha sido traducido a multitud de idiomas y se han desarrollado numerosos cursos en todo el mundo para fomentar su utilización. Epanet permite el trabajo habitual, con el uso de su interfaz gráfica y con un entorno de programación. Para esto último la EPA facilitó una biblioteca que permite a los usuarios desarrollar aplicaciones personalizadas.

### *1.3.1 Análisis hidráulico en Epanet*

El método que emplea EPANET para resolver simultáneamente las ecuaciones de continuidad en los nodos y las ecuaciones de comportamiento hidráulico de las tuberías, para un instante dado, puede clasificarse como un método híbrido de nodos y mallas. Todini y Pilati (1987), y más tarde Salgado(1988) decidieron llamarlo "Método del Gradiente".

En esta formulación, las ecuaciones individuales de energía en cada tubería se combinan con las ecuaciones individuales de conservación en cada nodo para proveer una solución simultánea tanto de cargas en los nodos como caudales individuales en las tuberías. Similar a lo realizado en los métodos

[1] **HydroLogic Systems BV.** http://www.hydronet.com.
[2] **Inc Bently Systems**. http://communities.bentley.com.
[3] **KYPipe.** http://www.kypipe.com.

"Circuito Simultáneo" y "Teoría Lineal", las ecuaciones no lineales de energía son linealizadas usando la expansión en series de Taylor. Sin embargo, a diferencia de métodos predecesores en este caso las ecuaciones son resueltas usando un eficiente esquema que emplea la inversión de la matriz de coeficientes originales (Rossman, 2000).

### *1.3.2 Análisis de la calidad del agua en Epanet*

Las ecuaciones que gobiernan el simulador que utiliza EPANET para determinar la calidad del agua, están basadas en el principio de conservación de la masa, acoplado con las cinéticas de reacción. Durante la dispersión de una sustancia no hay intercambio de masa entre porciones de agua adyacentes mientras éstas viajan por las tuberías. La concentración de una sustancia en el agua cuando abandona el nodo, es simplemente la suma ponderada respecto a los caudales, de las concentraciones de todos los flujos que llegan al nudo. Sin embargo, mientras el agua viaja por las tuberías, las sustancias disueltas pueden verse transportadas hasta la pared y reaccionar con materiales como los productos de la corrosión que se desarrolla en la misma la pared. Por lo que la variación de la concentración de una sustancia depende de las reacciones que ocurren en el interior de las tuberías o embalses. Luego, una sustancia disuelta en el agua es transportada a lo largo de la tubería con la misma velocidad media que el fluido, y al mismo tiempo reacciona creciendo o decreciendo con una cierta velocidad de reacción (Rossman, 2000).

## 1.4 Toolkit para sistemas de cómputo de modelación de redes de distribución.

Los requerimientos técnicos y operativos en la operación y gestión de las redes han aumentado considerablemente, y con ellos, se emplean nuevas herramientas específicas para cada problema particular. Para este fin, se han desarrollado diversas bibliotecas (toolkit) que sirven como motores del cálculo hidráulico y del análisis de la calidad del agua.

### *1.4.1 OOPnet*

En el año 2015 se presenta una nueva versión de OOPnet (D. Steffelbauer, y otros, 2015) un Epanet orientado a objeto escrito en python. La filosofía del diseño de OOPnet está en correspondencia con sus dos principales objetivos, usabilidad y cooperación. Este toolkit utiliza varias bibliotecas desarrolladas en python para facilitar y optimizar las diferentes funcionalidades requeridas en el toolkit. En su diseño se conciben tres funciones fundamentales: read, plot y run; cada una de ellas emplea una serie de paquetes que permiten realizar el flujo de trabajo completo de un usuario. En el momento que se desarrolla esta investigación aún no ha salido a la luz la biblioteca.

### *1.4.2 CWSnet*

En el año 2002 se propuso a CWSnet (M. Guidolin, y otros, 2002) como sustituto del toolkit de Epanet. Esta propuesta arrojaba resultados numéricamente comparables y ofrecía mejores o similares funcionalidades,

además de proveer un grado alto de extensibilidad y compatibilidad. CWSnet fue completamente desarrollado en C++ usando programación orientado a objeto. Internamente del CWSnet está dividido en tres capas: capa de red, la capa solucionador hidráulico y la capa de cálculos matemáticos. Sin embargo, esta biblioteca no realiza análisis de calidad del agua.

### *1.4.3 Porteau*

El Porteau (Piler, y otros, 2011) es otro toolkit orientado a objeto que realiza análisis de distribución de agua. Para el desarrollo del mismo se empleó Java para la capa del modelado de la red y para la interfaz gráfica de usuario. Los módulos de cálculos fueron codificados en C++ y se utilizó la Interfaz Nativa de Java para comunicarse con el jre (*Java Runtime Environnement*). La principal desventaja que presenta el mismo es que no realiza análisis de la calidad del agua.

### *1.4.4 Epanet C++*

El toolkit de Epanet (EPA, 2008) para programadores es una biblioteca de enlace dinámico (.dll) la cual contiene funciones que permite a desarrolladores adaptar el motor computacional de Epanet a sus necesidades específicas. La primera versión fue lanzada en 1993 y la última en febrero de 2008. El toolkit para programadores de Epanet es una extensión del paquete de simulación de EPANET. Las funciones del toolkit existen para poner todo el flujo de trabajo de un análisis bajo el control del programador. La *Fig. 1* muestra el flujo de datos de este toolkit.

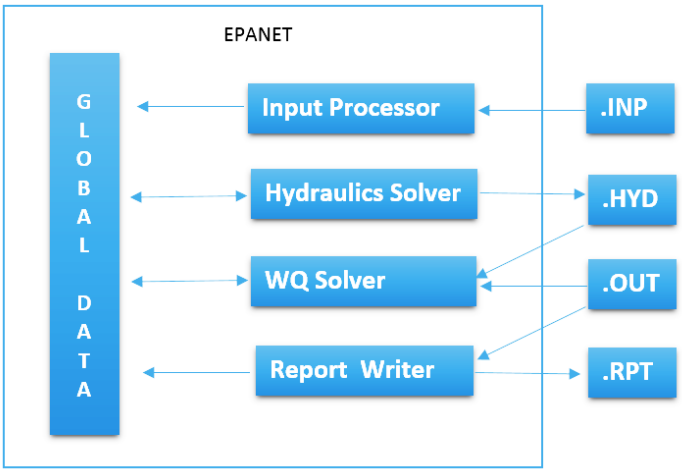



Fig. 1 Flujo de dato del toolkit de Epanet (EPA, 2008).

El módulo que procesa la entrada recibe un fichero con extensión .INP que contiene la descripción de la red. El contenido es analizado, interpretado y almacenado en el área de memoria compartida.

El módulo de solucionador hidráulico realiza las simulaciones de periodos extendidos. Los resultados en cada paso del tiempo pueden ser escritos en un archivo externo en formato binario (HYD).

Si una simulación de la calidad del agua es requerida, el Módulo de calidad del agua accede al flujo de datos desde el archivo hidráulico y este computa cómo la sustancia es transportada y cómo reacciona en la red con el paso del tiempo.

Durante este proceso pueden ser escritos ambos resultados, tanto el hidráulico como el de calidad del agua, en un fichero binario (.OUT).

Si se consulta algún reporte, el módulo correspondiente lee el resultado de los análisis realizados que se han almacenado en el fichero de salida (.OUT) y genera un reporte cuyos valores son almacenados en el fichero de reporte (RPT).

Desafortunadamente el diseño de este toolkit tiene algunas limitaciones que hacen de cualquier intento por extenderlo, adicionarle nuevas funcionalidades o mejorar las existentes un trabajo difícil de lograr y que consume mucho tiempo (M. Guidolin, y otros, 2002).

### *1.4.5 Epanet Java*

Originalmente los archivos del Toolkit de Epanet fueron escritos en C y se limitaban a sistemas operativos que pudieran interpretar archivos DLL. Sin embargo, en el año 2013 el proyecto AWARE-P (Baseform, 2013) publicó la reescritura completa del código fuente de Epanet en el lenguaje de programación Java. La biblioteca es de código abierto y permite entre otras cosas:

- Realizar simulaciones hidráulicas, de calidad y MSX.
- Exportar modelos hidráulicos en formatos INP, XML o archivos de Microsoft Excel.
- Editar una red y sus parámetros en Microsoft Excel e importarla de nuevo al programa.
- Resultados detallados paso por paso, nudo por nudo y tubería por tubería con precisión de 15 decimales y fácilmente exportables a Microsoft Excel.

De los toolkit antes mencionado se escoge el toolkit de Epanet en Java para realizar los cálculos hidráulico y el análisis de la calidad del agua, pues Epanet es considerado en la industria de software el estándar internacional para el modelado hidráulico, por otra parte al ser una rescritura del toolkit desarrollado en C y utilizar un paradigma orientado a objeto resuelve la principal limitación de este y se beneficia de sus potencialidades. Al contar con el código fuente de Epanet Java disponible en GitHub[4] podemos, una vez validado el sistema, subirlo para que sea incorporado y utilizado por la comunidad.

[4] https://github.com/Baseform/Baseform-Epanet-Java-Library

## 1.5 Incorporación de funcionalidades a Epanet Java

A partir de un análisis realizado al código de Epanet Java se identificaron dos modos de utilizar los cálculos hidráulicos y de calidad para lograr añadir funcionalidades a la biblioteca. El primero es utilizando la clase ENToolkit2 la cual contiene métodos que permiten consultar resultados de una simulación, este modo abstrae al programador del funcionamiento interno de la biblioteca. El segundo modo es heredando e instanciando objetos de las clases que realizan el cálculo hidráulico y el análisis de calidad, este modo logra una mayor integración con la biblioteca. Fue escogido el segundo modo pues este brinda más libertad al programador para realizar las simulaciones y trabajar con los resultados de estas. De modo que estaremos extendiendo Epanet Java para desarrollar la herramienta.

## 1.6 Biblioteca para modelar y resolver problemas de optimización

En el epígrafe *1.2.2* se mostró como el modelo matemático utiliza programación lineal entera para determinar los orígenes de contaminación. Para resolver este tipo de problema se han implementado soluciones en el lenguaje Java. En la documentación consultada se encontraron dos, una de ellas es la interfaz Java ILP[5] y la otra es la biblioteca JOM[6]. A efectos de uso (resolver un problema de programación lineal entera) no hay ventajas entre una u otra. Sin embargo, se selecciona JOM por la facilidad del formato en que devuelve los resultados. En JOM los resultados de cada variable se devuelven en forma de matriz y en ILP en un objeto de tipo String que luego hay que transformar para su posterior utilización.

JOM es una biblioteca de Java, gratis y de código abierto, que permite modelar y resolver problemas de optimización. Para la modelación es necesario definir los parámetros de entrada, las variables de decisión, la función objetivo, y las restricciones. Para solucionar los problemas de programación lineal entera, JOM permite elegir que solucionador utilizar, GLPK[7] o CPLEX[8]. Se escoge GLPK por ser libre a diferencia de CPLEX cuya licencia es propietaria.

## 1.7 Tecnologías de desarrollo.

### *1.7.1 Metodología de desarrollo.*

La metodología de desarrollo dirige el proceso de creación de los programas computacionales e incide de manera contundente en el éxito del producto. Existen dos enfoques fundamentales: metodologías pesadas y metodologías ágiles. Las primeras están pensadas para el uso exhaustivo de documentación generada durante todo el ciclo de vida de grandes proyectos que implican un gran número de desarrolladores. Mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios, la confianza

[5] http://javailp.sourceforge.net/
[6] http://www.net2plan.com/jom/
[7] https://www.gnu.org/software/glpk/
[8] http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/

en las habilidades del equipo y en mantener una buena relación con el cliente. Estas metodologías son utilizadas en equipos pequeños de desarrollo. El alcance y el número de desarrolladores de esta investigación descartan las metodologías pesadas. Dentro de las metodologías ágiles destacan XP y OpenUp. Se decide guiar el proceso de desarrollo de software utilizando OpenUp debido a que esta se centra en la arquitectura, lo cual es de vital importancia en el desarrollo de una extensión. Además es ligera, proporciona una comprensión detallada del proyecto, aplica enfoques iterativos e incrementales y define las fases, actividades y artefactos que se generan durante el ciclo de desarrollo del software.

### *1.7.2 Lenguaje de programación.*

Debido a la selección del toolkit de Epanet en Java se escoge a Java como lenguaje de programación. Java es un lenguaje de propósito general que utiliza el paradigma orientado a objeto, es multiplataforma y además permite la computación concurrente.

### *1.7.3 Lenguaje de modelado.*

El Lenguaje Unificado de Modelado (UML) prescribe una notación estándar y semánticas esenciales para el modelado de un sistema orientado a objetos. Un diseño orientado a objetos puede ser modelado con cualquiera de las docenas de metodologías populares que existen, pero causaría a los revisores tener que aprender las semánticas y notaciones de la metodología empleada antes que intentar entender el diseño en sí. Ahora, con el UML se prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos que describe la semántica esencial de lo que estos diagramas y símbolos significan. Por tales motivos se elige como lenguaje de modelado a utilizar UML.

### *1.7.4 Herramienta para automatización de pruebas.*

Realizar comprobaciones de forma manual sobre el código que se escribe, es un proceso en desuso debido a la complejidad del mismo. En la actualidad se han desarrollado herramientas que permiten automatizar esta tarea. Como el lenguaje de desarrollo seleccionado fue Java, se selecciona JUnit como framework para evaluar que el funcionamiento de cada uno de los métodos de las clases se comporta como se espera. Es decir, se utilizará para realizar pruebas de software de nivel unitario.

### *1.7.5 Entorno de desarrollo integrado.*

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) es un entorno de programación que ha sido empaquetado como un programa de aplicación, este cuenta con las potencialidades de interpretar, compilar, depurar y diseñar interfaces. Con su utilización los desarrolladores agilizan la creación de programas. Un IDE puede soportar varios lenguajes de programación o estar específicamente diseñado para solo uno.

Se selecciona como Entorno de Desarrollo Integrado a Intellij IDEA en su versión Community, para garantiza compatibilidad con la interfaz de usuario de Epanet Java. Intellij IDEA Community Edition[9] es la versión de código abierto de IntelliJ IDEA, un IDE Premier para Java, Groovy y otros lenguajes de programación como Scala o Clojure. Esta versión permite la integración de frameworks de Pruebas como JUnit y TestNG; también permite diseñar interfaces de usuario, utilizar sistemas de control de versiones, entre otras funcionalidades.

### *1.7.6 Herramienta CASE.*

Las herramientas de Ingeniería de Software Asistida por Computadoras (CASE, en sus siglas en Inglés), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas tales como: el proceso de realizar un diseño del proyecto, la generación automática de parte del código a partir de un diseño dado, entre otras. La herramienta CASE seleccionada para realizar los artefactos pertinentes a las etapas de análisis y diseño de la solución presentada es Visual Paradigm para UML en su versión 8.0.

## 1.8 Conclusiones del capítulo

El estudio realizado permitió conocer conceptos inherentes de los sistemas de modelación de redes de distribución hidráulica y de los procesos de la dinámica simplificada. Además, favoreció la discusión sobre las soluciones existentes y confirmó las ventajas de utilizar la biblioteca Epanet Java. Se seleccionó la biblioteca JOM para la modelación y cálculo de los problemas de optimización, así como al solucionador GLPK. Se analizó el modo en que Epanet realiza el análisis hidráulico y de calidad del agua y se arribó a la conclusión, que al transportar las sustancias con la misma velocidad que el fluido que atraviesa las tuberías, este valor puede ser utilizado para calcular el tiempo que demora en pasar un fluido a través de ellas, lo cual hace posible integrar a Epanet Java los procesos de la dinámica simplificada. Además, se definió el modo en que se utilizará Epanet Java para el análisis de contaminación estudiado. Por otro lado, las tecnologías de desarrollo y las herramientas seleccionadas permitirán crear los artefactos inevitables para lograr un sistema bien documentado.

[9] **JetBrains s.r.o.** https://www.jetbrains.com/idea

# Capítulo 2: Análisis y diseño del sistema

En el desarrollo de software es necesario realizar un análisis de las necesidades o requisitos del proceso que se desea informatizar; y diseñar una solución lógica que satisfaga los requisitos y las restricciones imperantes en el problema. "*La esencia del análisis y el diseño orientado a objeto consiste en situar el dominio de un problema y su solución lógica dentro de la perspectiva de los objetos"*. (Larman, 1999)

## 2.1 Análisis del dominio

El análisis orientado a objeto para el desarrollo de un sistema requiere identificar los conceptos, los atributos y las asociaciones relevantes en el dominio del problema. El resultado de este análisis se expresa en el modelo conceptual. Dicho modelo tiene como meta lograr un conocimiento básico del vocabulario y de los conceptos que se incluyen en los requerimientos. La *Fig. 2* muestra el modelo conceptual del problema que se estudia.

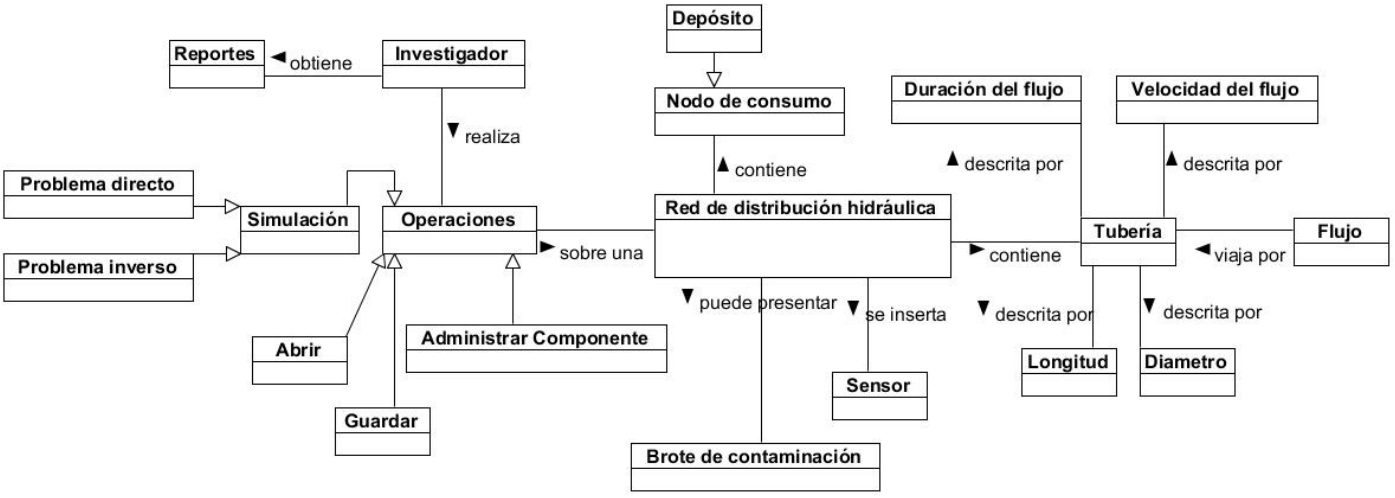



Fig. 2 Modelo conceptual

### *2.1.1 Descripción de los objetos del modelo conceptual.*

**Investigador:** Representa al usuario que realiza operaciones sobre una red de distribución hidráulica y que obtiene reportes de las simulaciones.

**Red de distribución hidráulica:** Instancia de un objeto que contiene la información necesaria de la red de distribución hidráulica que se estudia para el análisis de contaminación.

**Reportes:** Ventanas y ficheros en las cuales el investigador visualiza los resultados del análisis de contaminación.

**Operaciones:** Representa los procedimientos que el usuario debe poder hacer.

**Simulación:** Representa la automatización de un problema de la dinámica simplificada.

**Problema directo:** Es la ejecución de la distribución de un contaminante a través de la red.
**Problema inverso:** Es la ejecución del proceso de estimación del origen de contaminación.
**Abrir:** Es la operación de abrir un fichero previamente salvado desde la herramienta.
**Guardar:** Es la operación de guardar un fichero que contenga los datos de la red de distribución hidráulica.
**Administrar componente:** Es la operación de crear, visualizar, editar o eliminar algún componente de la red de distribución hidráulica.
**Brote de contaminación:** Representa la contaminación que se vierte en un nodo de la red.
**Sensor:** Representa el patrón de los tiempos de contaminación detectado en un nodo de la red.
**Nodo de consumo:** Lugar donde se almacena y consume el agua**.**
**Depósito:** Lugar donde se almacena agua, pero cuya capacidad de almacenamiento es limitada.
**Tubería:** Conducto por el que viaja el agua.
**Duración del flujo:** Tiempo que demora en pasar el flujo por la tubería.
**Velocidad del flujo:** Rapidez del flujo que viaja por la tubería.
**Longitud:** Extensión de la tubería.
**Diámetro:** Diámetro de la tubería.
**Flujo:** Movimiento de una sustancia liquida por la tubería.

## 2.2 Análisis de los requisitos

El objetivo de realizar análisis de requerimientos es identificar y documentar claramente la necesidad del cliente; de modo que no se presenten sorpresas al momento de entregar el producto. Por lo tanto, definirlos de manera inequívoca es de vital importancia pues ellos determinan los requisitos funcionales y no funcionales del sistema. (Larman, 1999)

La necesidad actual es poder emplear de manera automática el modelo matemático que resuelve el problema directo e inverso de la dinámica simplificada. Esto implica:

- Adicionar sensores y vertidos de contaminación en nodos de la red.
- Automatizar el proceso de análisis de contaminación del modelo matemático.
- Ver los resultados del análisis.
- El sistema debe poder ejecutarse en cualquier sistema operativo.

Como resultado de estos requerimientos se definen los siguientes requisitos funcionales y no funcionales del sistema.

### *2.2.1 Requisitos funcionales*

Los requisitos funcionales del sistema son las funciones que este debe realizar. Estos pueden ser agrupados según su acción en el sistema.

**CU 1 Administrar vertidos de contaminación.**

**RF1** Adicionar el vertido de contaminación en un nodo de la red.

**RF2** Modificar el vertido de contaminación en un nodo de la red.

**RF3** Eliminar el vertido de contaminación en un nodo de la red.

**RF4** Visualizar el vertido de contaminación en un nodo de la red.

**CU 2 Administrar sensores.**

**RF5** Adicionar sensor a un nodo de la red.

**RF6** Modificar el sensor de un nodo de la red.

**RF7** Eliminar el sensor de un nodo de la red.

**RF8** Visualizar el sensor de un nodo de la red.

**RF9 Ejecutar el problema directo de la dinámica simplificada**

Realiza la distribución de la contaminación por la red de distribución hidráulica.

**RF10 Ejecutar el problema inverso de la dinámica simplificada**

Realiza la estimación del origen de contaminación en la red de distribución hidráulica.

**RF11 Guardar mapa procesado.**

Crea y almacena un fichero con todos los elementos que compone el mapa de red procesado por la herramienta.

**RF12 Abrir fichero creado por la herramienta.**

Abre un fichero creado previamente por la herramienta.

**RF13 Visualizar reporte inferencia**.

Permite visualizar el reporte del análisis de contaminación del problema inverso.

**RF14 Visualizar reporte de contaminación.**

Permite visualizar el reporte del análisis de contaminación del problema directo

**RF15 Guardar reporte.**

Permite crear y almacenar un fichero que contiene la información de un reporte.

### *2.2.2 Requisitos no funcionales*

Los requisitos no funcionales del sistema son características o restricciones que debe cumplir.

**Requisitos de hardware**

Los requisitos de hardware mínimos para la ejecución del sistema son:

**RNF1** Procesador: Intel(R) Celeron(R) 1.80 GHz

**RNF2** RAM: 2 GB.

**RNF3** Disco Duro: 128 MB

**Requisitos de software**

Los requisitos de software mínimos para la ejecución del sistema son:

**RNF4** Máquina Virtual de Java 1.7

**RNF5** Biblioteca GLPK 4.8

## 2.3 Actores del sistema

Los actores del sistema son agentes externos al sistema que interactúan con él. En la *Tabla 1* se describe el actor de la herramienta.

Tabla 1 Actor del Sistema

| Actor | Descripción |
|---|---|
| Usuario | Persona que interactúa con la herramienta computacional. |

## 2.4 Casos de uso del sistema

El caso de uso es un documento que describe la secuencia de eventos de un agente externo que interactúa con el sistema durante la ejecución de un proceso. Los casos de uso pueden agrupar requisitos funcionales del sistema en las historias que narran.

### *2.4.1 Diagrama de Casos de Uso del Sistema*

En la *Fig. 3* se presenta el diagrama de caso de uso del sistema, el cual representa la interacción del actor con los casos de usos.

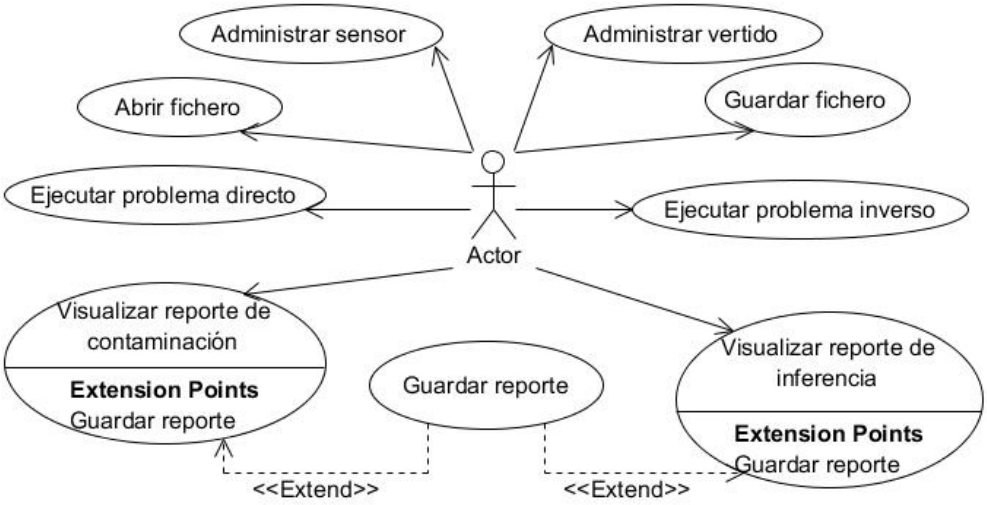



Fig. 3 Diagrama de caso de uso de la herramienta computacional.

En la *Fig. 3* se puede notar el uso de patrones de casos de uso, los cuales son empleados generalmente como plantillas que describen cómo deberían ser estructurados y organizados los casos de uso. Los patrones usados fueron CRUD total y extensión concreta. La aplicación de ellos permitió describir las relaciones y dependencias de las acciones que deben poder realizarse desde la herramienta computacional.

**CRUD total:** Su nombre es un acrónimo de las palabras en inglés Create, Read, Update, Delete. Se aplica cuando en un caso de uso se requiere mostrar, modificar, adicionar o eliminar información. Ejemplos: Caso de uso Administrar sensor y Administrar vertido.

**Extensión concreta:** Este patrón especifíca como el comportamiento de un caso de uso base puede incorporar el comportamiento de otro caso de uso.

### *2.4.2 Descripción de los casos de uso del sistema*

A continuación se pueden observar algunas descripciones de los casos de uso que narran como los usuarios interactúan con el sistema para estimar el origen de contaminación a través de la herramienta computacional. En la *Tabla 2* se expone como adicionar, eliminar, editar o visualizar los sensores de la red.

Tabla 2 Descripción del caso de uso del sistema Administrar sensor.

| | |
|---|---|
| **Objetivo** | Administrar sensor |
| **Actores** | Usuario (inicia) |
| **Resumen** | El caso de uso se inicia cuando el usuario selecciona adicionar, editar o eliminar un sensor. Finaliza cuando se muestra en pantalla el resultado de la acción sobre el sensor. |

| | |
|---|---|
| **Complejidad** | Media |
| **Prioridad** | Crítica. |
| **Referencias** | CU1 |
| **Precondiciones** | Debe ser visualizado un mapa de la red. |
| **Poscondiciones** | Se visualiza el resultado de la acción sobre el sensor. |

**Flujo de eventos**

**Flujo básico < Adicionar sensor >**

| | **Actor** | **Sistema** |
|---|---|---|
| 1. Selecciona la opción adicionar sensor.<br><br>3. Inserta los datos solicitados en el formulario. | | 2. Muestra un formulario que recoge los siguientes datos del sensor: nodo, tiempo de inicio, paso y tiempo de activación. Así como los instantes en los que se detectó contaminación.<br>4. Almacena los datos del sensor.<br>5. Colorea de verde los nodos con sensor. |

**Flujo alterno**

**Cancelar la adición del** sensor

| | **Actor** | **Sistema** |
|---|---|---|
| 3. Selecciona la opción cancelar. | | 4. El sistema cierra la ventana. |

**Flujo básico < Edición de un sensor >**

| | **Actor** | **Sistema** |
|---|---|---|
| 1. Selecciona la opción editar sensor.<br><br>3. Selecciona el nodo donde se encuentra el sensor a editar.<br><br>5. Modifica los datos del sensor. | | 2. Muestra una lista que permite al usuario seleccionar el sensor que desea editar.<br>4. Carga los datos del sensor: nodo, tiempo de inicio, paso y tiempo de activación. Así como los instantes en los que se detectó contaminación.<br>6. Almacena los datos del sensor.<br>7. Colorea de verde los nodos con sensor. |

**Flujo alterno**

| | | |
|---|---|---|
| **Cancelar la edición del sensor** | | |
| | **Actor** | **Sistema** |
| 5. Selecciona la opción cancelar. | | 6. El sistema cierra la ventana. |
| **Flujo básico <Eliminación de un sensor >** | | |
| | **Actor** | **Sistema** |
| 1. Selecciona la opción eliminar sensor.<br>3. Selecciona el sensor a eliminar.<br>5. Confirma la eliminación. | | 2. Muestra una lista que permite al usuario seleccionar el sensor que desea eliminar.<br>4. Envía un mensaje de confirmación.<br>6. Elimina el sensor de la memoria.<br>7. Colorea de negro el nodo que tenía el sensor. |
| **Flujo alterno** | | |
| **Cancelar la eliminación de un sensor** | | |
| | **Actor** | **Sistema** |
| 5. Cancela la eliminación del sensor. | | 6. Cierra el formulario. |
| **Flujo básico <Visualizar sensores >** | | |
| **Actor** | | **Sistema** |
| 1. Selecciona la opción visualizar sensor. | | 2. Colorea de verde todos los nodos con sensor. |
| **Relaciones** | **CU Incluidos** | No incluye otros casos de uso. |
| | **CU Extendidos** | No extiende otros casos de uso |
| **Requisitos no funcionales** | No es restringido por ningún requisito no funcional. | |
| **Asuntos pendientes** | No se considera la realización de ningún asunto de este tipo. | |

En la *Tabla 3* se describe la interacción entre el usuario y el sistema durante el proceso de ejecución de inferencia del origen de contaminación.

Tabla 3 Descripción de caso de uso del sistema Ejecutar problema inverso.

| | | |
|---|---|---|
| **Objetivo** | Ejecutar problema inverso de la dinámica simplificada | |
| **Actores** | Usuario (inicia) | |
| **Resumen** | El caso de uso se inicia cuando el usuario selecciona la opción de ejecutar el análisis de contaminación del problema inverso de la dinámica simplificada. Finaliza cuando se notifica al usuario si la acción fue realizada de manera satisfactoria. | |
| **Complejidad** | Alta | |
| **Prioridad** | Crítica. | |
| **Referencias** | RF10. | |
| **Precondiciones** | Debe ser visualizado un mapa de la red. | |
| **Poscondiciones** | Se pueden obtener reportes del análisis de contaminación. | |
| **Flujo de eventos** | | |
| **Flujo básico < Ejecutar problema inverso>** | | |
| | **Actor** | **Sistema** |
| | 1. Selecciona la opción de Ejecutar el problema inverso.<br><br>3. Inserta el nombre del análisis de contaminación.<br><br>7. Da clic en el botón Aceptar. | 2. Muestra una ventana que permite insertar el nombre del análisis de contaminación.<br><br>4. Almacena el nombre del análisis de contaminación.<br><br>5. Ejecuta el análisis y muestra un mensaje avisando al usuario que esto tardará unos minutos.<br><br>6. El sistema colorea los probables orígenes de contaminación de color azul. Si no se encuentra solución el sistema lo notifica al usuario.<br><br>8. Cierra la ventana de notificación. |
| **Flujo alterno** | | |

| **Solución no encontrada** | |
|---|---|
| Actor | Sistema |
| 3. Inserta el nombre del análisis de contaminación.<br><br>7. Da clic en el botón Aceptar. | 4. Almacena el nombre del análisis de contaminación.<br><br>5. Ejecuta el análisis de contaminación y muestra un mensaje avisando al usuario que esto tardará unos minutos.<br><br>6. El sistema realiza la estimación y si no encuentra solución lo notifica al usuario.<br><br>8. Cierra la ventana de notificación. |
| **Flujo alterno** | |
| **Cancelar el análisis de contaminación del problema inverso** | |
| Actor | Sistema |
| 5. Cancela la ejecución del análisis de contaminación. | 6. Cierra la ventana de administración del análisis. |

| | | |
|---|---|---|
| **Relaciones** | **CU Incluidos** | No incluye otros casos de uso. |
| | **CU Extendidos** | No extiende otros casos de uso |
| **Requisitos no funcionales** | No es restringido por ningún requisito no funcional. | |
| **Asuntos pendientes** | No se considera la realización de ningún asunto de este tipo. | |

## 2.5 Arquitectura del sistema

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, por lo que indica la estructura, funcionamiento e interacción entre las partes del sistema informático (Garlan, y otros, 1994). La arquitectura seleccionada para el desarrollo de la herramienta computacional fue N-capas. Este patrón

arquitectónico logra alcanzar separación e independencia entre los elementos del sistema permitiendo el desarrollo incremental. En la herramienta computacional se definen 2 capas, en la *Fig. 4* se puede observar las relaciones que se establecen entre los diferentes paquetes de las capas:

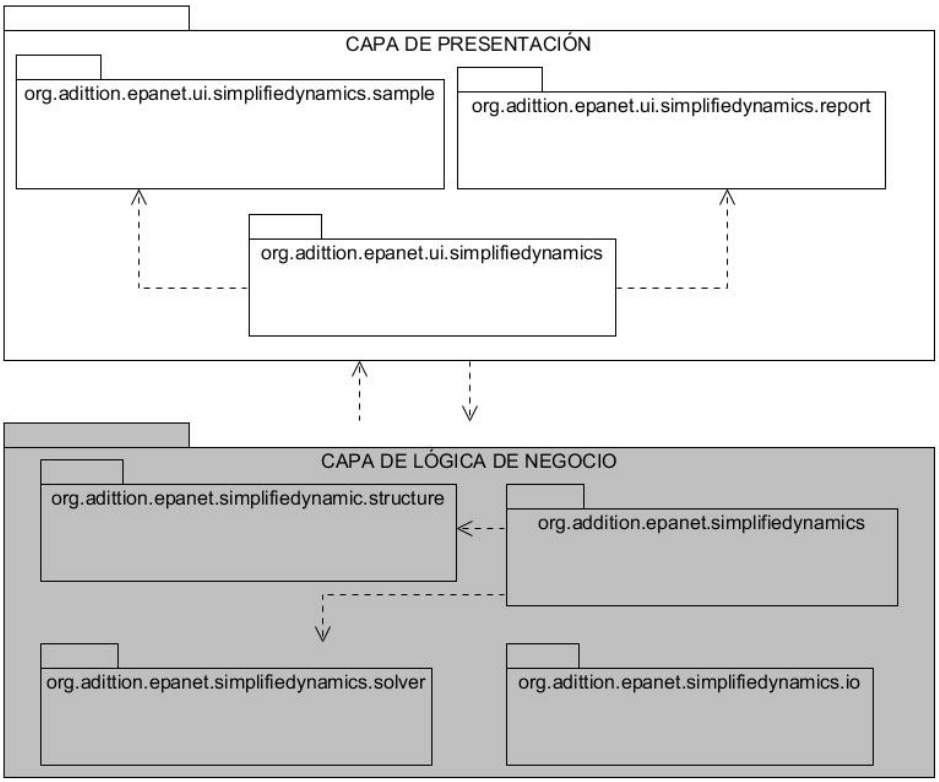



Fig. 4 Arquitectura de la herramienta computacional.

**Capa de presentación:** En esta capa se agrupan todas las clases que permiten la visualización e interacción con el usuario. A través de ella se obtienen los parámetros de entrada y se observa la salida que arroja el análisis de contaminación. En la herramienta computacional, la capa de presentación contiene los paquetes org.adittion.epanet.ui.simplifiedynamics, org.adittion.epanet.ui.simplifiedynamics.report y org.adittion.epanet.ui.simplifiedynamics.sample.

**Capa de lógica de negocio:** En esta capa se agrupan todas las clases que permiten realizar el procesamiento de la información capturada en la capa de presentación. A través de ella se logra realizar la ejecución de las simulaciones. En la herramienta computacional, la capa de lógica de negocio contiene los paquetes org.adittion.epanet.simplifiedynamic.structure, org.adittion.epanet.simplifiedynamics.solver, org.adittion.epanet.simplifiedynamics.io y org.addition.epanet.simplifiedynamics.

## 2.6 Diagrama de clases

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contiene: clases, asociaciones y atributos, interfaces con sus operaciones y constantes, métodos, información sobre los tipos de atributo, navegabilidad y dependencia. La *Fig. 5* muestra el diagrama de clases de la herramienta computacional.

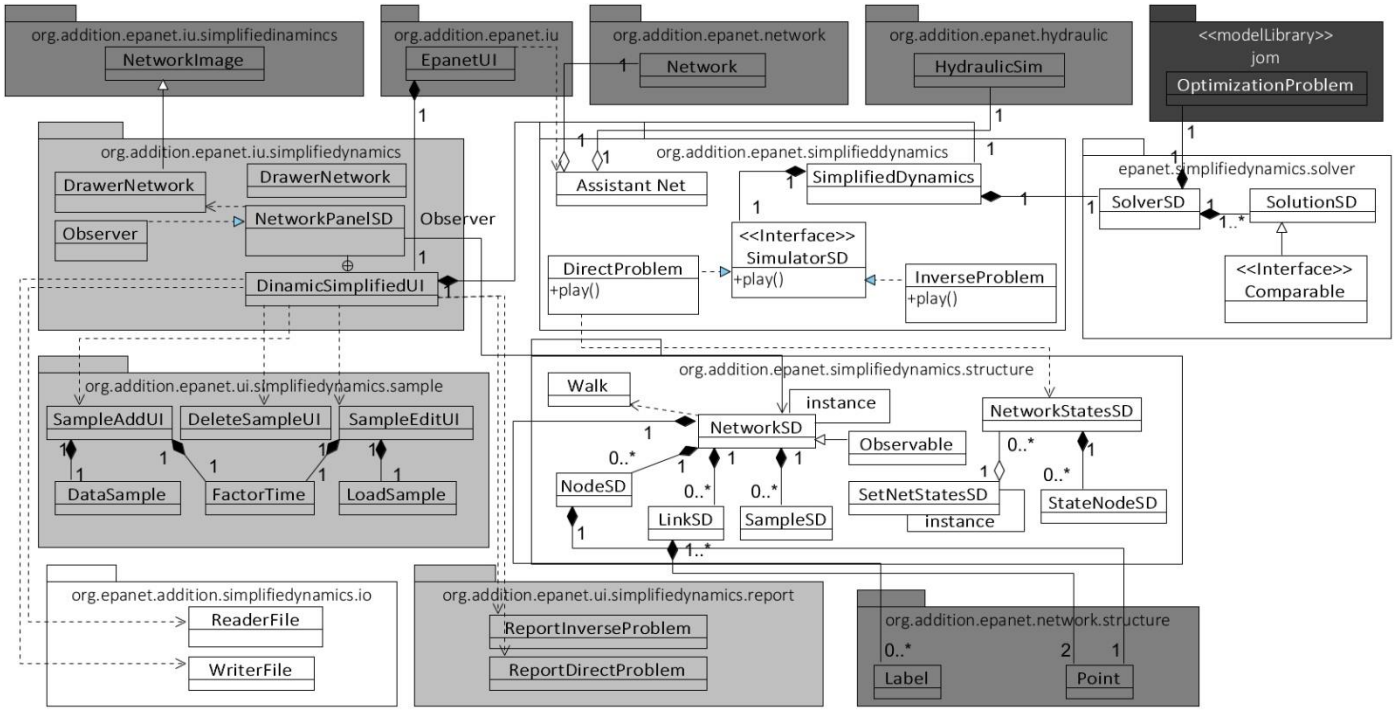



Fig. 5 Diagrama de clase de la herramienta computacional

### *2.6.1 Patrones de diseño*

Los patrones de diseño son soluciones generales a problemáticas comunes de diseño que se presentan en el desarrollo de software. La aplicación de un patrón permite, a través de una serie de pasos, adaptar el sistema a una estructura definida lo cual garantiza optimalidad y una arquitectura robusta.

Los patrones GRASP por sus siglas en inglés: *General Responsibility Assignment Software Patterns* o Patrones Generales de Software para Asignación de Responsabilidades, nos indican cómo asignar responsabilidades a los objetos del sistema. En la solución fueron usado los patrones: experto en información, creador, bajo acoplamiento y alta cohesión.

**Experto en información**: el patrón "experto en información" nos sugiere que debemos asignar las responsabilidades a aquellos objetos (o clases de objetos) que disponen de la información para hacerlo. Este patrón es usado en todas las clases. Por ejemplo, cada clase posee los métodos que permiten cambiar u obtener sus atributos.

**Creador:** el patrón "Creador" nos invita a discutir quien es el encargado de crear un determinado objeto. En la solución está presente por ejemplo en la clase SolverSD que es la responsable de crear una instancia de la clase SolutionSD utilizada para resolver el problema inverso.

**Bajo acoplamiento:** Cuando se habla de acoplamiento entre objetos, se hace referencia a la "fuerza" con la que ciertos objetos están relacionados, o dependen unos de otros. Mientras más dependencia tenga un objeto de otros para llevar a cabo sus tareas, más fuerte será el acoplamiento. Cuando una clase de objetos puede realizar sus tareas, sin depender de ninguna otra clase de objetos (o de un número muy reducido de ellas), se dice que hay bajo acoplamiento.

**Alta cohesión:** Cada clase de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de las clases y que tribute a estas facilidades para realizar sus tareas. A su vez, las clases no deben ser saturadas de métodos que pueden ser delegados a otras clases. Por ejemplo, en el diagrama de clase propuesto, la clase NetworkSD define varios métodos usando la información de la cual ella es experta que auxilian a otras clases a realizar su responsabilidad.

Otro conjunto de patrones muy utilizados en el diseño de sistema son los patrones Gof. Los patrones de la "pandilla de los cuatro" (Gang-of-Four) se expresan como una forma indispensable de enfrentarse a la programación a raíz del libro “Design Patterns—Elements of Reusable Software” (Gamma, y otros, 1994). Estos se pueden clasificar en seis categorías: fundamentales, de creación, de partición, estructurales, de comportamiento y de concurrencia (Grand, 1998). En la solución propuesta se emplearon los siguientes patrones: Singleton (de creación), Observer( de comportamiento), Delegation ( fudamental).

**Delegation**: Es un patrón que permite a una clase reutilizar el comportamiento de otra clase sin heredar de ella. Se utiliza en el diseño de la herramienta computacional al modelar las clases que definen el procedimiento del análisis de contaminación de un problema de la dinámica simplificada. A decir, ProblemaSD(clase que delega), DirectProblem, InverseProblem y la interfaz simulatorSD. Garantizando así la reutilización del código, pues de este modo, al desear incorporar un nuevo procedimiento del análisis de contaminación solo se requiere crear una clase que implemente la interfaz simulatorSD.

**Singleton**: Es un patrón que garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. Este patrón es usado en la herramienta computacional al definir las clases NetworkSD y SetNetStatesSD.

**Observer:** Es un patrón de comportamiento utilizado cuando interesa saber el estado de un objeto (observado) y notificarlo a otros objetos (observadores) para que también ellos cambien su estado. Por

ejemplo en la herramienta computacional es aplicado a las clases NetworkSD (objeto observado) y NetworkPanelSD (Observador).

## 2.7 Procedimiento para la automatización del análisis de contaminación

Como se pretende automatizar los problemas directo e inverso, es necesario describir el proceso para llevar a cabo estas tareas durante la implementación. Tanto para el problema directo como para el problema inverso los primeros pasos son:

1. Ejecutar una simulación hidráulica.
2. Visualizar el mapa de la red.

Luego se realizan los siguientes pasos según el problema que se quiera resolver. Para el análisis de contaminación del problema directo se procederá de la siguiente manera:

3. Distribuir la contaminación a partir de los brotes de contaminación.
4. Devolver resultados del análisis de contaminación.

Para el análisis de contaminación del problema inverso se procederá del siguiente modo:

3. Obtener el grafo final $G_f' \left(V_f', E_f'\right)$ a partir del patrón de tiempo detectado en el(los) sensor(es).
4. Modelar el grafo final $G_f' \left(V_f', E_f'\right)$ como un problema de optimización lineal.
5. Resolver el problema de optimización.
6. Devolver resultados del análisis de contaminación.

En el primer paso es donde se ejecuta la simulación hidráulica en la que Epanet calcula la velocidad y el flujo que viaja a través de las tuberías y se crea una instancia de la clase NetworkSD que almacena los valores a partir de los resultados del análisis de contaminación. Luego, como siguiente paso, la clase DrawerNetwork dibuja la red de distribución.

En el problema directo, el tercer paso es distribuir la contaminación a partir de la ecuación (1.1) según los brotes de contaminación insertados por el usuario. En este paso se considera el tiempo que demora cada fluido en pasar a través de la tubería y se construye una instancia de la clase NetworkStatesSD que es un grafo bipartido, donde el conjunto de nodos Y representan los nodos $S_i^t$ orígenes de contaminación y un conjunto de nodos X que representa los nodos $S_j^t$ que fueron contaminados respectivamente por cada $S_i^t$. Para representar los nodos de este grafo se utilizará la clase StateNodeSD. El cuarto paso, es devolver el patrón detectado en un nodo sobre el que el usuario de un clic.

Por otro lado, el tercer paso del problema inverso consiste en obtener el grafo final. Para lograr esto, primero se almacenan los nodos $S_i^t$ que son observados como no contaminados y todos aquellos que están enlazados con ellos (nodos blancos). Luego se comprueba que los nodos observados como contaminados no están en contacto con ningún otro nodo previamente observado como contaminado ni que sea un nodo

blanco. De este modo se construye una instancia de la clase NetworkStateSD que representa el grafo final. El próximo paso es modelar este grafo final como un problema de optimización, para esto se utiliza la clase SolverSD quien también es la responsable de resolver el problema de optimización utilizando la clase OptimizationProblem de la biblioteca JOM. Las soluciones encontradas se almacenan en una lista que es devuelta a la clase SimplifiedDynamics. Finalmente se colorean de azul aquellos nodos que son probables orígenes de contaminación.

## 2.8 Conclusiones del capítulo

Durante la fase de inicio y elaboración del sistema, siguiendo la metodología OpenUp, se realizó el análisis y diseño de la solución. Se dio especial atención a la obtención de los requisitos del sistema y a su arquitectura. Se obtuvieron artefactos que proporcionan una comprensión clara y detallada del sistema que se desea construir; y que favorecen el inicio de la implementación.

El uso de los patrones de diseños permitió dar una solución eficiente y elegante a problemas de diseños presentados. Además, se propuso un procedimiento para la automatización del análisis de contaminación mediante el problema directo e inverso de la dinámica simplificada. Por lo tanto, en la fase de implementación del sistema se debe emplear fielmente el diseño y el procedimiento propuesto, para así lograr un sistema capaz de cumplir con las expectativas del cliente y con normas de calidad.

# Capítulo 3: Implementación y pruebas

En este capítulo se describen las pautas tomadas en la implementación y se detallan las pruebas del software realizadas. Además, se presentan los diferentes componentes de la herramienta. Finalmente se valida la solución.

Los artefactos generados durante el análisis y diseño son la entrada fundamental del flujo de trabajo de implementación. En esta fase se implementa la solución y se organiza el código utilizando las pautas, la arquitectura y los patrones definidos en las fases anteriores.

## 3.1 Estándares de codificación

Los estándares de codificación son convenciones que nos indican las reglas para aplicar un estilo y formato coherente en todo el código. En la presente solución se utilizan los estándares de codificación del lenguaje Java presentados en *Java Language Specification* (Gosling, y otros, 2011).

### *3.1.1 Convención de nombres*

Las convenciones de nombres hacen los programas más entendibles haciéndolos más fácil de leer. En la *Tabla 4* se describen las reglas usadas para nombrar a los archivos de código fuente, clases, atributos, variables, métodos entre otros.

Tabla 4 Descripción de la convención de nombres

| Identificador | Convención | Ejemplo |
|---|---|---|
| Paquetes | El prefijo del nombre de un paquete se escribe siempre con letras ASCII en minúsculas, y debe ser uno de los nombres de dominio de alto nivel.<br>Los subsecuentes componentes del nombre del paquete variarán de acuerdo a las convenciones de nombres internas de cada organización. | org.addition.epanet.simplifieddynamics |
| Archivo fuente | El nombre de la clase y el fichero son el mismo. | InverseProblemSD.java |
| Clase o estructura | Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Utiliza UpperCamelCase[10]. | class InverseProblemSD |

[10] CamelCase es un estilo de escritura que se aplica a frases o palabras compuestas. Existen dos tipos de CamelCase:

- UpperCamelCase, cuando la primera letra de cada una de las palabras es mayúscula.
- lowerCamelCase, igual que la anterior con la excepción de que la primera letra es minúscula.

| | | |
|---|---|---|
| Interface | Los nombres de las interfaces siguen la misma regla que las clases. | Interface SimulatorSD |
| Variable | Los nombres deben ser descriptivos y cortos. Utiliza lowerCamelCase[10]. | idNode;<br>selectedNode; |
| Métodos | La primera palabra debe ser verbo. Utiliza lowerCamelCase[10]. | addNode() |
| Constantes | Los nombres de las variables declaradas como constantes deben ir totalmente en mayúscula. Si es compuesto se separa con un subguión (_). | Static final int<br>MAX_ALLOWED_LENGTH = 20; |

### *Estilo de código*

Las convenciones de estilo de código mejoran la lectura de software permitiendo entender el código mucho mejor. Estas pueden ser clasificadas en convenciones de organización de ficheros, convenciones de márgenes, convenciones de comentarios, convenciones de declaraciones y de sentencias. A continuación se describen las convenciones de estilo de código aplicadas durante la implementación

**Organización de ficheros:** Cada fichero fuente Java contiene una única clase o interface pública. Cuando algunas clases o interfaces privadas están asociadas a una clase pública, pueden ponerse en el mismo fichero que la clase pública. La clase o interfaz pública es la primera clase o interface del fichero. Los ficheros tienen la siguiente estructura:

- Comentarios de comienzo.
- Sentencias package e import.
- Declaraciones de clases e interfaces.

**Márgenes:**

- Se emplean cuatro espacios como unidad de indentación.
- Se evitan las líneas de más de 80 caracteres.
- Cuando una expresión no entre en una línea, se rompe de acuerdo con estos principios:
  · Romper después de una coma.
  · Romper antes de un operador.
  · Roturas de alto nivel (más a la derecha que el "padre").

**Comentarios:**

- Se usan comentarios de implementación delimitados por /*…*/, y //.
- Los ficheros fuente comienzan con un comentario en el que se lista el nombre de la clase, información de la versión, fecha, y copyright.

**Declaraciones:**

- Se escriben declaración por línea, ya que facilita los comentarios.
- Se inicializar las variables locales donde se declaran, menos que el valor inicial dependa de algunos cálculos que deben ocurrir luego.
- Poner las declaraciones solo al principio de los bloques.

**Sentencias:**

- Cada línea debe contener como mucho una sentencia.
- Una sentencia *return* con un valor no debe usar paréntesis.

## 3.2 Diagrama de componentes

El diagrama de componente contienen las partes físicas de un sistema y la relación que hay entre ellas. Este diagrama estructura el modelo de implementación en componentes, de ahí que un componente puede ser la implementación de más de una clase.

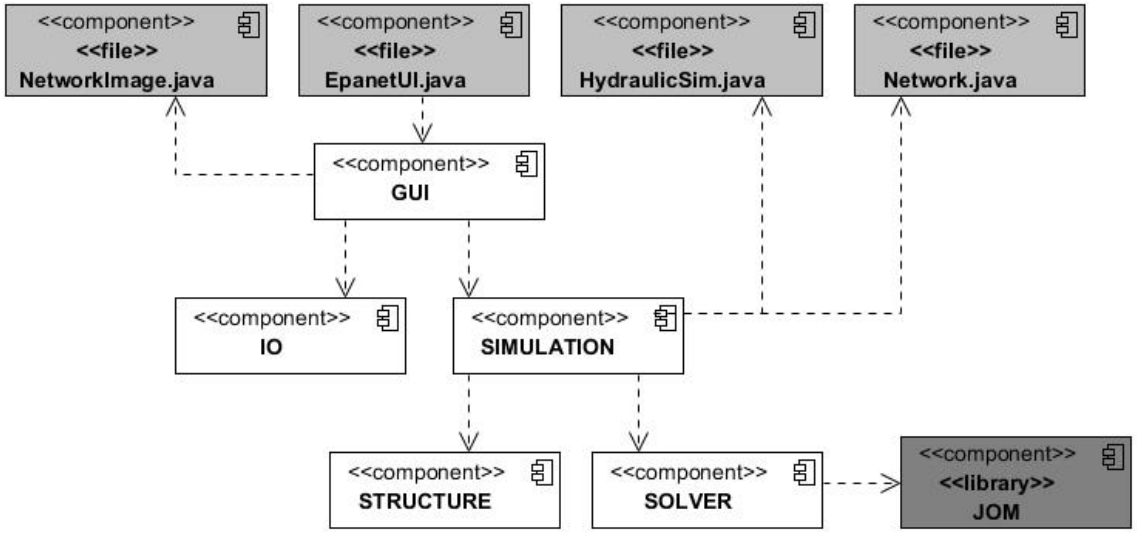



Fig. 6 Diagrama de Componente de la herramienta computacional.

**EpanetUI.java:** Principal interfaz gráfica de Epanet Java.

**HydraulicSim.java:** Clase de Epanet Java que ejecuta una simulación hidráulica.

**Network.java:** Clase de Epanet Java que contiene toda la información de la red de distribución hidráulica.

**NetworkImage.java:** Clase de Epanet Java que dibuja la red de distribución hidráulica.

**GUI:** Componente que contiene las interfaces de usuario.

**IO:** Componente que agrupa las clases que procesan los ficheros de entrada y salida.

**SIMULATION:** Componente que agrupa las clases que permiten el análisis de contaminación.

**STRUCTURE:** Componente que contiene las clases que representan los elementos de la red.

**SOLVER:** Componente que agrupa las clases que obtienen todas las soluciones de la inferencia de contaminación.

**JOM:** Biblioteca que resuelve problemas de programación lineal entera.

## 3.3 Pruebas de software

Según (Pressman, y otros, 2015) las pruebas se pueden ver como un procedimiento secuencial. Primeramente las pruebas deben centrarse en las partes más pequeñas del software, que en el contexto orientado a objeto son las clases. A continuación se deben ir integrando los módulos hasta que se conforma el sistema completo. Luego se realizan pruebas de alto nivel donde se comprueban los criterios de validación establecidos durante el análisis de requisitos. En la herramienta computacional se seguirá la estrategia clásica realizando pruebas unitarias y pruebas de sistema.

### *3.3.1 Pruebas unitarias*

Las pruebas unitarias tienen como objetivo comprobar el buen funcionamiento de las componentes más elementales del código, por ejemplo los métodos no triviales de una clase. En la solución propuesta se utiliza JUnit para automatizar las pruebas unitarias. En la *Fig.7* se muestra el resultado obtenido después de aplicar esta prueba a los métodos walkFrom y walkTo de la clase NetworkSD.

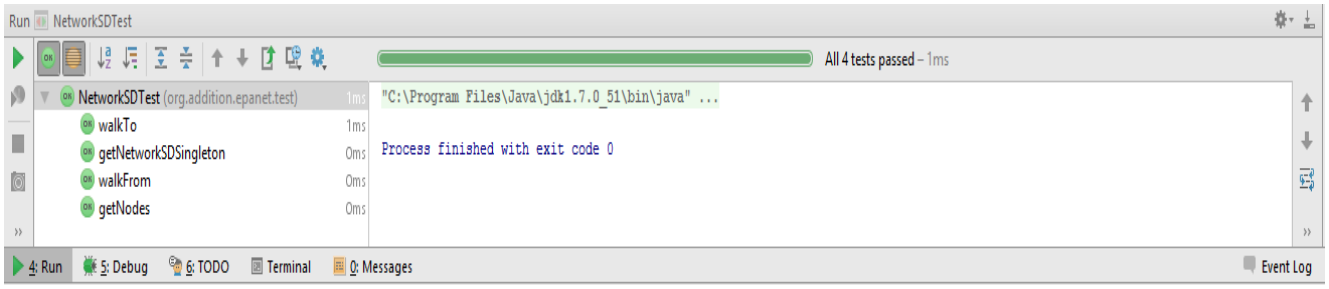



Fig. 7 Prueba realizada a los métodos walkFrom y walkTo utilizando JUnit.

### *3.3.2 Resultado de las pruebas unitarias*

En este nivel de prueba se detectaron dos no conformidades en el método *play()* de la clase *InverseProblemSD* pues este no construía correctamente el grafo final y en el método *recursiveSolution(SolutionSD Xo, weightOF)* de la clase *SolverSD*, el cual no encontraba todas las posibles soluciones. Estas no conformidades fueron registradas y corregidas posteriormente.

### *3.3.3 Pruebas de sistema*

Los casos de prueba a diseñar en este nivel, cubren los aspectos funcionales y no funcionales del sistema. Por lo tanto, se tiene presente los requisitos iniciales y la descripción de casos de uso. En la presente

solución se realizan pruebas funcionales utilizando siete mapas[11] de ciudades (Ver *Tabla 5*). Estos mapas son utilizados por los investigadores en sus estudios. El mapa de la ciudad de Modena, Italia, fue el utilizado en (Ortega Díaz, 2015) para comprobar el método desarrollado. Los Anexos 2 y 3 revelan casos de pruebas aplicados al caso de uso Administrar sensor.

Tabla 5 Mapas de redes de distribución tomadas como casos de estudio.

| Nombre del mapa | Cantidad de nodos | Cantidad de tuberías |
|---|---|---|
| Gesler1985 | 12 | 14 |
| Nytun | 20 | 21 |
| Anytown | 25 | 46 |
| Modena | 272 | 317 |
| Richmond_standard | 872 | 957 |
| Wolf | 1786 | 1995 |
| Exnet | 1894 | 2467 |

**Caso de uso:** Administrar sensor.

**Descripción general:** El caso de uso se inicia cuando el usuario selecciona adicionar, visualizar, editar o eliminar un vertido. Finaliza cuando se muestra en pantalla el resultado de la acción sobre el vertido.

**Condiciones de ejecución**: El usuario debe haber ejecutado la herramienta computacional.

Tabla 6 Secciones a probar del caso de uso del sistema Administrar sensor.

| Nombre de la sección | Escenario | Descripción |
|---|---|---|
| SC 1: Adicionar sensor | EC 1.1: Adicionar sensor correctamente | Después de acceder al formulario e insertar los datos. El sistema colorea el nodo con sensor de color verde. |
| | EC 1.2: Adicionar sensor sin entrar datos. | Después de acceder al formulario para insertar los datos, el usuario deja uno de los campos vacíos. Luego el sistema debe emitir un mensaje notificando el error. |
| | EC 1.3: Cancelar la adición de un sensor. | Después de acceder al formulario para insertar los datos, el usuario cancela la operación. Luego el sistema debe cerrar la ventana. |

[11] http://emps.exeter.ac.uk/engineering/research/cws/downloads/benchmarks/

| | | |
|---|---|---|
| | EC 1.4: Introducir texto en campos numéricos. | Después de acceder al formulario para insertar los datos, el usuario inserta texto en campos numéricos. Luego el sistema debe borrar el texto insertado. |
| | EC 1.5: Introducir datos incoherentes. | Después de acceder al formulario para insertar los datos, el usuario inserta datos incoherentes. Se consideran datos incoherentes aquellos que no corresponden con las restricciones de los campos (Ver *Tabla 7*). Luego el sistema debe emitir un mensaje notificando el error. |
| SC 2: Editar sensor | EC 2.1: Editar sensor correctamente. | Después de acceder al formulario, el sistema muestra un campo de selección de nodos con sensores. Seguidamente el usuario selecciona el identificador del nodo donde se encuentra el sensor y el sistema responde cargando los datos. Luego de editar los datos el sistema colorea el nodo con sensor de color verde. |
| | EC 2.2: Editar sensor sin entrar datos. | Después que el sistema carga los datos del sensor seleccionado, el usuario deja algún campo vacío. Luego el sistema debe emitir un mensaje notificando el error. |
| | EC 2.3: Cancelar la adición de un sensor. | Después que el sistema carga los datos del sensor seleccionado, el usuario cancela la operación. Luego el sistema debe cerrar la ventana. |
| | EC 2.4: Introducir texto en campos numéricos. | Después que el sistema carga los datos del sensor seleccionado, el usuario inserta texto en campos numéricos. Luego el sistema debe emitir un sonido y borrar el texto insertado. |
| | EC 2.5: Introducir datos incoherentes. | Después que el sistema carga los datos del sensor seleccionado, el usuario inserta datos incoherentes. Se consideran datos incoherentes aquellos que no corresponden con las restricciones de los campos. (Ver *Tabla 7)* Luego el sistema debe emitir un mensaje notificando el error. |
| SC 3: Eliminar sensor | EC 3.1: Eliminar sensor correctamente | Después que el sistema carga los datos de todos los sensores de la red, el usuario selecciona uno de ellos y presiona el botón eliminar. Luego el sistema elimina el sensor |

| | | |
|---|---|---|
| | | de la lista de sensores. Finalmente, el usuario cierra la ventana. |
| | EC 3.2: Cancelar la eliminación de un sensor. | Después que el sistema carga los datos de todos los sensores de la red, el usuario cancela la operación. El sistema debe cerrar la ventana. |
| SC 4: Visualizar sensor | EC 4.1: Visualizar todos los sensores. | Después que el usuario selecciona la opción de visualizar el sistema colorea todos los nodos con sensores de color verde. |

Tabla 7 Variables de las secciones adicionar y editar del caso de uso del sistema Administrar sensor.

| No | Nombre del campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | Id Node | Campo de Selección | Vacío | Permite seleccionar el nodo al que se le desea adicionar un sensor. |
| 2 | Time Start (min) | Campo de número | Vacío | Permite insertar el minuto del análisis de contaminación en que el que se activó el sensor. |
| 3 | Time Step (min) | Campo de número | Vacío | Permite insertar cada cuántos minutos el sensor tomó muestras. Su valor debe ser menor que el campo Time Duration. |
| 4 | Time Duration (min) | Campo de número | Vacío | Permite insertar el total de minutos que el sensor estará activado. |
| 5 | Time (min) | Campo de número | Vacío | Permite adicionar los tiempos en los que el sensor detectó contaminación. |

**Caso de Uso:** Ejecutar problema inverso de la dinámica simplificada.

**Descripción general:** El caso de uso se inicia cuando el usuario selecciona la opción de ejecutar un análisis de contaminación del problema inverso de la dinámica simplificada. Finaliza cuando se muestra los resultados del análisis de contaminación.

**Condiciones de ejecución:** El usuario debe haber ejecutado la herramienta computacional.

Tabla 8 Sección a probar del caso de uso del sistema Ejecutar problema inverso.

| Nombre de la sección | Escenario | Descripción |
|---|---|---|
| SC 1: Ejecutar simulación del problema inverso | EC 1.1: Ejecutar el análisis de contaminación del problema inverso correctamente | Después de haber definido todos los nodos con sensores, el usuario ordena ejecutar el problema inverso. Luego el sistema notifica al usuario que el análisis de contaminación se está ejecutando. Cuando finaliza la ejecución el sistema muestra una ventana con los resultados. |
| | EC 1.2: Ejecutar el análisis de contaminación del problema inverso sin sensores. | Después de haber ejecutado la herramienta computacional, el usuario ordena ejecutar el problema inverso sin haber definido al menos un sensor. Luego el sistema notifica al usuario que debe insertar al menos un sensor. |
| | EC 1.3: Guardar resultado del análisis de contaminación. | Después que el sistema muestra una ventana con los probables orígenes de contaminación, el usuario selecciona la opción guardar de dicha ventana. Luego, el sistema almacena el reporte en el mismo directorio donde se ejecuta la herramienta computacional, en un fichero con formato csv. |

**Caso de Uso:** Ejecutar problema directo de la dinámica simplificada.

**Descripción general:** El caso de uso se inicia cuando el usuario selecciona la opción de ejecutar un análisis de contaminación del problema directo de la dinámica simplificada. Finaliza cuando se notifica al usuario si la acción fue realizada de manera satisfactoria.

**Condiciones de ejecución:** El usuario debe haber ejecutado la herramienta computacional.

Tabla 9 Secciones a probar del caso de uso del sistema Ejecutar problema directo.

| Nombre de la sección | Escenario | Descripción |
|---|---|---|
| SC 1: Ejecutar análisis del | EC 1.1: Ejecutar el análisis del problema directo correctamente. | Después de haber definido todos los nodos con brotes de contaminación, el usuario ordena ejecutar el problema directo. El sistema solicita al usuario un identificador para |

| | | |
|---|---|---|
| problema directo | | el análisis de contaminación. Luego el sistema notifica al usuario que ha concluido el análisis. |
| | EC 1.2: Ejecutar el análisis de contaminación del problema directo sin brotes de contaminados. | Después de haber ejecutado la herramienta computacional, el usuario ordena ejecutar el problema directo sin haber definido al menos un nodo con brote de contaminación. Luego el sistema notifica al usuario que debe insertar al menos un brote de contaminación en algún nodo. |
| | EC 1.3: Visualizar resultado del análisis de contaminación del problema directo. | Después que el sistema notifica al usuario que ha concluido el análisis de contaminación de manera correcta, el usuario da clic sobre uno de los nodos de la red pintada en el panel de visualización. Luego el sistema muestra una ventana con los tiempos en los que el nodo fue contaminado. |
| | EC 1.4: Guardar resultado de la contaminación en un nodo. | Después que el sistema muestra una ventana con los tiempos en los que un nodo fue contaminado, el usuario selecciona la opción guardar de dicha ventana. Luego, el sistema almacena el reporte en el mismo directorio donde se ejecuta la herramienta computacional, en un fichero con formato csv. |

### *3.3.4 Resultado de las pruebas de sistema*

En OpenUp se realizan varias iteraciones de pruebas con el objetivo de validar la calidad del sistema. En el caso de la herramienta computacional fue necesario realizar dos iteraciones de pruebas durante la fase de transición. En la *Tabla 10* se muestran las no conformidades encontradas y en la *Fig. 8* el comportamiento de las no conformidades encontradas en cada iteración.

Tabla 10 Estado de las no conformidades encontradas durante las pruebas de sistema.

| Caso de Uso | Sección | Escenario | Estado |
|---|---|---|---|
| Administrar Sensor | 1 | EC 1.2: Adicionar sensor sin entrar datos. | Solucionado |
| | | EC 1.5: Introducir datos incoherentes. | Solucionado |
| | 2 | EC 2.5: Introducir datos incoherentes. | Solucionado |

| | | | |
|---|---|---|---|
| Ejecutar problema directo de la dinámica simplificada | 1 | EC 1.2: Ejecutar el análisis de contaminación del problema directo sin brotes de contaminados. | Solucionado |
| Ejecutar problema inverso de la dinámica simplificada | 1 | EC 1.1: Ejecutar el análisis de contaminación del problema inverso correctamente | Solucionado |
| | | EC 1.2: Ejecutar el análisis de contaminación del problema inverso sin sensores. | Solucionado |
| Abrir mapa | 1 | ESC 1.3 Abrir mapa de la red con formato inválido | Solucionado |

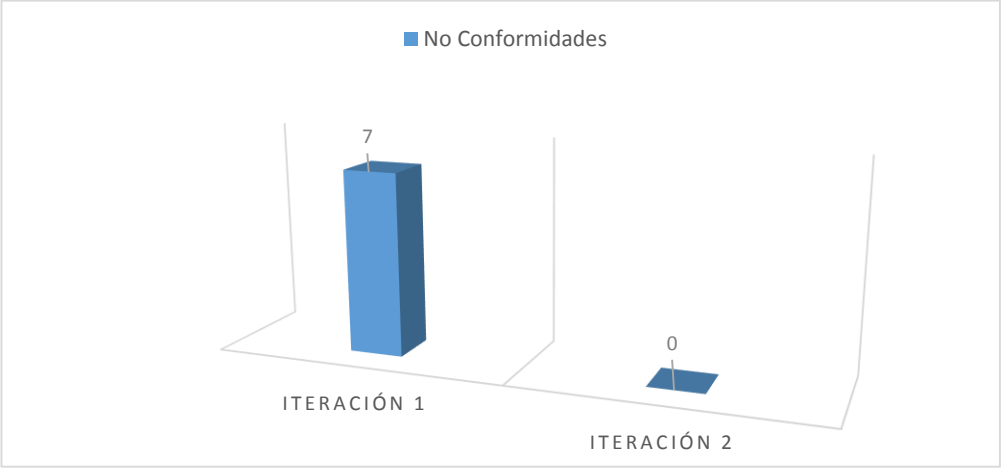



Fig. 8 Resultados de las pruebas de sistema a la herramienta computacional.

## 3.4 Validación de la solución propuesta

En (Ortega Díaz, 2015) se define la eficiencia del modelo matemático para el problema inverso como el número de veces en el cual se infiere el patrón de contaminación real y su nodo. Para la validación de la herramienta se utiliza el mismo criterio. Primeramente se simula el problema directo, distribuyendo la contaminación por toda la red. Luego se salva el patrón de contaminación detectado en uno o varios nodos. En un segundo momento se inserta un sensor con la información almacenada previamente y se resuelve el problema inverso. Este proceso se realiza con varios mapas y en distintas situaciones. En la *Tabla 11* se muestran algunos resultados obtenidos con el mapa de Modena, empleado en (Ortega Díaz, 2015) para la comprobación del modelo, y con Exnet un mapa de mayor tamaño.

Tabla 11 Comparación de resultados de análisis de contaminación en varios mapas.

| No | Mapa | Cant. nodos | Cant. tuberías | Cant. sensores | Tiempo Sensor 1 activado (min) | Tiempo Sensor 2 activado (min) | Tiempo Sensor 3 Activado (min) | Tiempos de contaminación. | Tiempo de ejecución. |
|---|---|---|---|---|---|---|---|---|---|
| 1. | Modena | 272 | 317 | 2 | 160 | 160 | - | 2 | 7s |
| 2. | Modena | 272 | 317 | 2 | 160 | 160 | - | 3 | 50s |
| 3. | Modena | 272 | 317 | 2 | 160 | 160 | - | 5 | 16 min. |
| 4. | Modena | 272 | 317 | 3 | 160 | 160 | 160 | 5 | 10 min |
| 5. | Exnet | 1894 | 2467 | 2 | 600 | 100 | - | 2 | 15s |
| 6. | Exnet | 1894 | 2467 | 3 | 200 | 200 | - | 3 | 50s |
| 7. | Exnet | 1894 | 2467 | 3 | 200 | 200 | 600 | 3 | 15s |
| 8. | Exnet | 1894 | 2467 | 3 | 200 | 200 | 600 | 10 | 20s |
| 9. | Exnet | 1894 | 2467 | 3 | 200 | 200 | 600 | 3 | 1 min |

No se realizó un estudio de eficiencia vs cantidad de nodos, pues las características de cada red son únicas y el número de nodos es fijo, pero sí se hizo un estudio de eficiencia vs duración de la fuente de contaminación. En el estudio se comprobó que, la eficiencia disminuye y el tiempo en encontrar las soluciones aumenta, en la medida que aumenta los tiempos de contaminación (Observe las tres primeras filas de la *Tabla 11*). Esto es consistente con el hecho que el método de optimización propuesto asume que la contaminación es un evento raro (Ortega Díaz, 2015). También se compararon los resultados que se obtienen según la cantidad de sensores y su ubicación en la red, se pudo observar que el método devuelve un conjunto de soluciones más reducido (que incluye el origen real) cuando se utiliza más de un sensor (Ver *Anexo 4*) y que una buena ubicación del sensor influye en la disminución del tiempo de ejecución (Compare las filas tres y cuatro, siete y nueve de la *Tabla 11*). En los Anexos 4, 5 y 6 se muestran distintas situaciones de pruebas y los resultados que devolvió la herramienta; los círculos rojos indican la presencia de un nodo origen de contaminación, los círculos verdes indican los nodos con sensores y los azules son los probables nodos orígenes de contaminación. Note que los orígenes reales siempre se encuentran en la solución.

## 3.5 Conclusiones del capítulo

Durante la implementación se utilizó el diagrama de clases propuesto en el capítulo anterior y se escribió el código bajo las normas de codificación del lenguaje Java. El uso de estándares de codificación facilitó la comprensión del código durante todo el desarrollo. Finalmente se validó la herramienta basándose en las pruebas de software y en el criterio de (Ortega Díaz, 2015). Las pruebas unitarias y de sistema aplicadas a la herramienta permitieron encontrar no conformidades en el código. Las no conformidades encontradas fueron sometidas a análisis y rectificación; una vez resueltas garantizaron obtener un sistema con mayor calidad. Los resultados obtenidos con la herramienta correspondieron a los esperados.

# Conclusiones generales

Luego de realizar un estudio de los conceptos fundamentales y de los sistemas informáticos existentes en el campo de acción de la investigación, se dio cumplimiento al objetivo planteado con la culminación de una herramienta informática para el análisis de la distribución de contaminación en redes de distribución hidráulica mediante los problemas de la dinámica simplificada, que cumple satisfactoriamente con los requisitos establecidos. Los resultados obtenidos durante la validación muestran que el comportamiento del modelo implementado es similar al encontrado en la bibliografía, lo cual demuestra la validez de la implementación.

Al integrarse a la biblioteca Epanet Java con un diseño adaptable a los posibles cambios, posibilita que se puedan agregar nuevas funcionalidades en versiones posteriores sin hacer grandes modificaciones; esto aumenta la utilidad de la herramienta informática con vistas a confeccionar un producto cada vez más completo y eficaz. Además, permite poner el código a disposición de la comunidad de Epanet lo cual aportaría otra fuente de validación de la solución desarrollada.

# Recomendaciones

Incorporar análisis de contaminación que considere la variación de la demanda en los nudos de consumo utilizando *“Belief propagation”.*

Diseñar un algoritmo para encontrar todos los posibles conjuntos de nodos que explican la contaminación, haciendo uso de programación paralela.

Realizar estudios para identificar las mejores ubicaciones de sensores en la red.

# Referencias Bibliográficas

**Agency, Enviroment Protection. 2005.** *Water distribution system analysis.* USA : s.n., 2005.

**Amendola diuana, Fabio y Pereira Ogawa, Seiti Caio Contardo. 2015.** *Análise comparativa dos modelos hidráulicos Epanet, WaterCad e Sistema de abastecimiento de àgua-rede de distribução.* Rio de Janeiro, Brasil : s.n., 2015.

**Baseform. 2013.** *http://www.baseform.org.* [En línea] Septiembre de 2013. [Citado el: 23 de Septiembre de 2015.] http://www.baseform.org/np4/epanetTool.html.

**Bently System, Inc. 2006.** *Bently Watergems v8i edition. User Guide.* 2006.

**Braunstein, A., Lage-Castellano, A. y Ortega, E. 2016.** *Contamination source inference in water distribution networks.* 2016.

**Butera, Ilaria, y otros. 2012.** *Recovering the release history of a pollutant intrusion into a water supply system through a geostatistical approach.* 2012.

**Cheung, P. B., Van Zyl, J. E. y Reis, L. F. R.** *Extension of Epanet for pressure driven demand modeling in water distribution system.*

**D. Steffelbauer y D. Fuchs-Hanusch. 2015.** OOPNET: An object-oriented EPANET in Python. *Institute of Urban Water Management and Landscape Water Engineering.* 2015.

**EPA.** EPA US Environmental Protection Agency. [En línea] [Citado el: 5 de 10 de 2015.] https://www.epa.gov/water-research/epanet.

**—. 2008.** EPANET Programmer's Toolkit. *Environmental Protection Agency's National Risk Management Research Laboratory.* 2008.

**Free Software Foundation, Inc.** [En línea] [Citado el: 20 de 11 de 2015.] https://www.gnu.org/software/glpk/.

**Gamma, Erich, y otros. 1994.** *Design Patterns—Elements of Reusable Software.* USA : Addison-Wesley, 1994. 0-201-63361-2.

**García Alcaraz, María del Mar. 2006.** *MODELACIÓN Y SIMULACIÓN DE REDES HIDRÁULICAS A PRESIÓN MEDIANTE HERRAMIENTAS INFORMÁTICAS.* Cartagena : s.n., 2006.

# Referencias Bibliográficas


**Garlan, David y Shaw, Mary . 1994.** *An introduction to Software Architecture.* New Jersey : V.Ambriola and G.Tortora, World Scientific, 1994.

**Gosling, James, y otros. 2011.** *The Java™ Language Specification Java SE 7 Edition.* [pdf] USA : s.n., 2011.

**Grand, Mark. 1998.** *Patterns in Java (volume 1).* USA : Wiley Computer, 1998. 978-0-471-22729-8..

**Grang, Mark. 1999.** *Patterns in Java (volume 2).* USA : Wiley Computer, 1999.

**Guan, Jiabao, y otros. 2006.** *Identification of contaminant sources in water distribution systems using simulation-optimization method: case study.* 2006.

**Iglesias Rey, Pedro L. .** [En línea] [Citado el: 5 de 10 de 2015.] http://www.instagua.upv.es/Epanet/.

**Larman, Craig. 1999.** *UML y Patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* México : Prentice Hall, 1999. 970-17-0261-1.

**M. Guidolin, P. Burovskiy y D.A. Savić. 2002.** CWSNet: An object-oriented toolkit for water distribution system simulations. *University of Exeter.* 2002.

**Mays, L., Ed, W. y Hill, McGraw. 2000.** *Water distribution system handbook-Introduction.* New York : s.n., 2000.

**Méndez Flores, Emmanuel. 2014.** Universo. *Métricas de cohesión y acoplamiento del software.* [En línea] 26 de 5 de 2014. [Citado el: 10 de 1 de 2016.] http://www.uv.mx/universo/563/infgral/infgral_36.html.

**Methods, Haestad. 2004.** *Advanced water distribution modeling and management.* Waterbury, USA : s.n., 2004.

**Molpeceres, Alberto. 2001.** *Convenciones de Código para el lenguaje de programación JAVA™.* [pdf] 2001.

**Ormsbee, Lindell E. 2006.** The history of water distribution network analysis: the computer age. s.l. : ASCE, 2006.

**Ortega Díaz, Ernesto. 2015.** *Inferencia de contaminación en redes distribución de fluido.* La Habana : s.n., 2015.

**Pardo, M. A., y otros.** *Iniciación a la programaciòn de un software de simulaciòn hidràulica.*

**Pavón Mariño, Pablo. 2013.** *Introducción a Java Optimization Modeler.* Cartagena, España : s.n., 2013.

**—.** Net2Plan. [En línea] [Citado el: 19 de 11 de 2015.] http://www.net2plan.com/jom/.

**Piler, Oliver, y otros. 2011.** PORTEAU: AN OBJECT-ORIENTED PROGRAMMING HYDRAULIC TOOLKIT FOR WATER DISTRIBUTION SYSTEMS ANALYSIS. *Cemagref, UR REBEX.* 2011.

**Pressman, Roger S. y Maxim, Bruce R. 2015.** *Software Engineering a practitioner´s approach Eighth edition.* New York : McGraw-Hill Education, 2015. 978-0-07-802212-8.

**Protato, Marco, Tryby, Michael y Piller, Olivier. 2007.** *Linear Algebra Analysis for Contaminant Source Indentification in Water Distribution Systems.* Tampa, Florida, USA : s.n., 2007.

**Rossman, Luis A. 2000.** EPANET 2 user's manual. *United States Environmental Protection Agency.* Cincinnati, U.S.A. : s.n., 2000.

**Saldarriaga, Juan G. 2007.** *Hidráulica de tuberìas- Abastecimiento de Agua, Redes, Riegos.* Bogotà : Alfaomega-Uniandes, 2007.

**Salomons, Elad y Ostfeld, Avi. 2010.** *Identification of possible contamination sources using reverse hydraulic simulation.* Arizona, USA : s.n., 2010.

**Sommerville, Ian. 2011.** *Software Engineering. Ninth Editioin.* s.l. : Pearson International Computer Science Series, 2011. 978-0-13-705346-9.

**Tiodini, E y Pilati, S. 1988.** *A gradient algorithm for the analysis of pipe networks.* 1988.

Univeristy of exeter. [En línea] [Citado el: 10 de 4 de 2016.] http://emps.exeter.ac.uk/engineering/research/cws/downloads/benchmarks/.

**Vega Niño, Oscar T. 2012.** *Herramienta de ayuda a la sectorización de redes de abastecimiento de agua basadas en la teorìa de grafos aplicando distintos criterios.* Valencia, España : s.n., 2012.

**Visual Paradigm, Team.** Visual Paradigm. User guide. [En línea] [Citado el: 18 de 2 de 2016.] https://www.visual-paradigm.com/.

# Anexos

## Anexo 1 Casos de prueba de los casos de uso del sistema abrir y guardar mapa.

| Escenario | Mapa de Red (File) | Respuesta del sistema | Resultado de la prueba | Flujo central |
|---|---|---|---|---|
| **ESC 1.1** Ejecutar la herramienta computacional | V Cargado | Epanet abre una nueva ventana que muestra la dirección del flujo del mapa de la red previamente cargado. | Satisfactorio | Epanet muestra un botón (Simplified Dynamics) que se habilita si hay un mapa procesado. Una vez accionado el botón, el sistema ejecuta una simulación hidráulica. Los valores de los parámetros de la red son procesados y se muestra una nueva ventana que contiene el mapa de la red. |
| **ESC 1.1** Ejecutar la herramienta computacional | I No cargado | Epanet muestra el botón (Simplified Dynamics) deshabilitado. | Satisfactorio | Epanet muestra un botón (Simplified Dynamics) deshabilitado. |
| **ESC 1.2** Guardar mapa de la red. | V Cargado | La herramienta computacional crea y almacena un fichero que contiene los datos de todos los elementos de la red. | Satisfactorio | Después de ser accionado el botón guardar de la herramienta computacional, el sistema crea y almacena un fichero que contiene los datos de todos los elementos de la red procesada. |
| **ESC 1.3** Abrir mapa de la red con formato correcto. | V Cargado | La herramienta computacional abre el fichero que contiene los datos de todos los elementos de la red. | Satisfactorio | Después haber cargado un mapa, el usuario acciona el botón abrir de la herramienta computacional. En seguida el sistema abre una ventana que permite buscar el mapa deseado. Después de ser seleccionado el mapa la herramienta procesa el fichero y |

| | | | | |
|---|---|---|---|---|
| | | | | muestra el mapa en el panel de dibujo. |
| **ESC 1.3**<br>Abrir mapa de la red con formato inválido | I<br>No cargado | La herramienta computacional muestra un mensaje que notifica al usuario del error ocurrido. | No satisfactorio | Después de haber cargado un mapa, el usuario acciona el botón abrir de la herramienta computacional. En seguida el sistema abre una ventana que permite buscar el mapa deseado. Luego de ser seleccionado el mapa la herramienta procesa el fichero, al detectar algún error muestra un mensaje que notifica al usuario del error ocurrido. |

**Anexo 2 Casos de pruebas de las secciones adicionar y editar sensor del caso de uso Administrar sensor.**

| Escenario | Id Node | Time Start (min) | Time Step (min) | Time Duration (min) | Time (min) | Respuesta del sistema | Resultado de la prueba | Flujo central |
|---|---|---|---|---|---|---|---|---|
| EC 1.1: Adicionar sensor correcta-mente | V<br>1 | V<br>0 | V<br>2 | V<br>20 | V<br>2,6,8,16,18 | El sistema colorea el nodo 1 de color verde. | Satisfacto-rio | Después de acceder al formulario e insertar los datos. El sistema colorea el nodo con sensor de color verde. |
| EC 1.2: Adicionar sensor sin entrar datos. | V<br>1 | I<br>Vacío | I<br>Vacío | I<br>Vacío | I<br>Vacío | El sistema notifica al usuario que ha dejado campos vacíos. | No Satisfacto-rio | Después de acceder al formulario para insertar los datos, el usuario deja uno de los campos vacíos. Luego el sistema debe emitir un mensaje notificando el error. |
| EC 1.3: Cancelar la adición de un sensor. | V<br>1 | I<br>Vacío | I<br>Vacío | I<br>Vacío | I<br>Vacío | El sistema cierra la ventana. | Satisfacto-rio | Después de acceder al formulario para insertar los datos, el usuario cancela la operación. Luego el sistema debe cerrar la ventana. |
| EC 1.4: Introducir texto en campos | V<br>2 | I<br>Cinco | I<br>Uno | I<br>Diez | I<br>Uno, tres, Siete. | El sistema va borrando todas las letras. | Satisfacto-rio | Después de acceder al formulario para insertar los datos, |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| numéricos al adicionar un sensor. | | | | | | | | el usuario inserta texto en campos numéricos. Luego el sistema debe borrar el texto insertado. |
| EC 1.5: Introducir datos incoherentes al adicionar un sensor. | V<br>105 | V<br>2 | I<br>30 | V<br>20 | I<br>10,15,40. | El sistema notifica al usuario que ha insertado datos incoherentes. | No satisfactorio. | Después de acceder al formulario para insertar los datos, el usuario inserta datos incoherentes. Se consideran datos incoherentes aquellos que no corresponden con las restricciones de los campos. Luego el sistema debe emitir un mensaje notificando el error. |
| EC 2.1: Editar sensor correctamente. | V<br>1 | V<br>0 | V<br>3 | V<br>30 | V<br>3,6,9,27 | El sistema almacena los datos y colorea el nodo con sensor de color verde. | Satisfactorio | Después de acceder al formulario, el sistema muestra un campo de selección de nodos con sensores. Seguidamente el usuario selecciona el identificador del nodo donde se encuentra el sensor y el sistema responde cargando los |

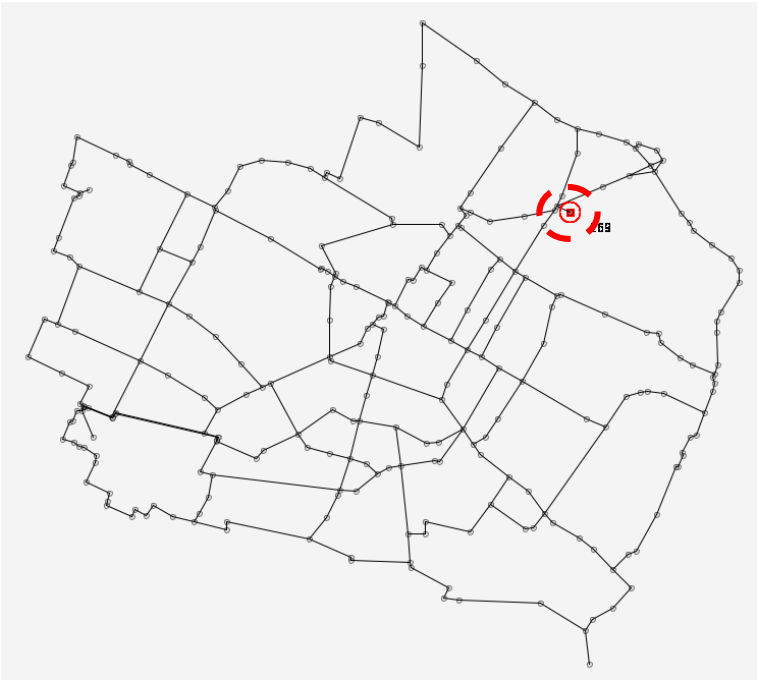
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | datos. Luego de editar los datos el sistema colorea el nodo con sensor de color verde. |
| EC 2.2: Editar sensor sin entrar datos. | V<br>1 | I<br>Vacio | I<br>Vacio | I<br>Vacio | I<br>Vacio | El sistema notifica al usuario que ha dejado campos vacíos. | Satisfacto-rio | Después que el sistema carga los datos del sensor seleccionado, el usuario deja algún campo vacío. Luego el sistema debe emitir un mensaje notificando el error. |
| EC 2.3: Cancelar la edición de un sensor. | V<br>9 | I<br>Vació | I<br>Vacío | I<br>Vacío | I<br>Vacío | El sistema cierra la ventana. | Satisfacto-rio | Después que el sistema carga los datos del sensor seleccionado, el usuario cancela la operación. Luego el sistema debe cerrar la ventana. |
| EC 2.4: Editar sensor introducien do textos en campos numéricos. | V<br>A | I<br>Tres | V<br>5 | V<br>20 | V<br>5,10,20 | El sistema borra todo el texto del campo Time Start. | Satisfacto-rio | Después que el sistema carga los datos del sensor seleccionado, el usuario inserta texto en campos numéricos. Luego el sistema debe emitir un sonido y borrar el texto insertado. |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| EC 2.5: Introducir datos incoheren-tes. | V<br>23 | V<br>20 | V<br>3 | I<br>1 | V<br>3,6,9 | El sistema muestra una ventana notificando al usuario que ha introducido datos incoherente s. | No Satisfacto-rio | Después que el sistema carga los datos del sensor seleccionado, el usuario inserta datos incoheren-tes. Se consideran datos incoheren-tes aquellos que no corresponden con las restricciones de los campos. Luego el sistema debe emitir un mensaje notificando el error. |

## Anexo 3 Casos de prueba de la sección eliminar sensor del caso de uso Administrar sensor.

| Escenario | Sensor | Respuesta del sistema | Resultado de la prueba | Flujo central |
|---|---|---|---|---|
| **EC 3.1:** Eliminar sensor correctamente | V 0_1,1,2,2,20 | El sistema elimina el sensor seleccionado. | Satisfactorio | Después que el sistema carga los datos de todos los sensores de la red y los muestra en un listado, el usuario selecciona uno de ellos y presiona el botón eliminar. Luego el sistema elimina el sensor de la lista de sensores. Finalmente, el usuario cierra la ventana. |
| **EC 3.2:** Cancelar la eliminación de un sensor. | V 1_10,10,2,2,20 | El sistema cierra la ventana de eliminación de sensores. | Satisfactorio | Después que el sistema carga los datos de todos los sensores de la red, el usuario selecciona un sensor pero finalmente decide cancelar la operación. El sistema debe cerrar la ventana. |

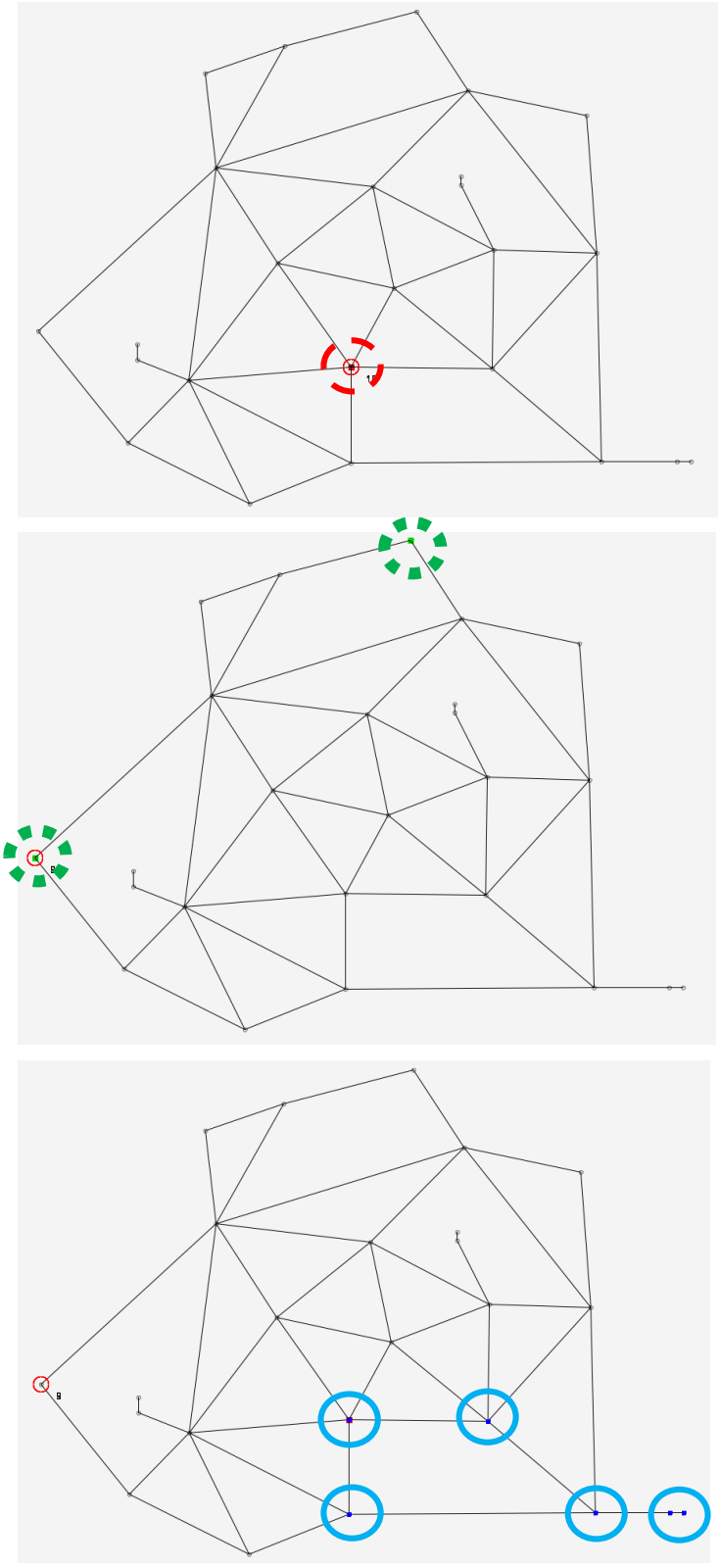
**Anexo 4 Análisis de inferencia en Modena.inp con uno y con dos sensores.**

Leyenda:

Sensor  Probable origen de contaminación  Origen de contaminación

## Anexo 5 Análisis de inferencia en anytown.inp con dos sensores.



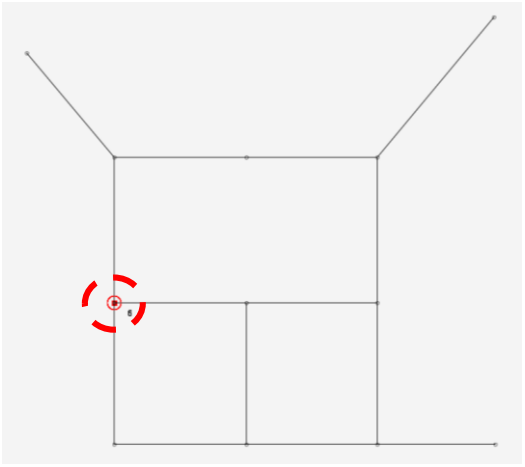
Leyenda:

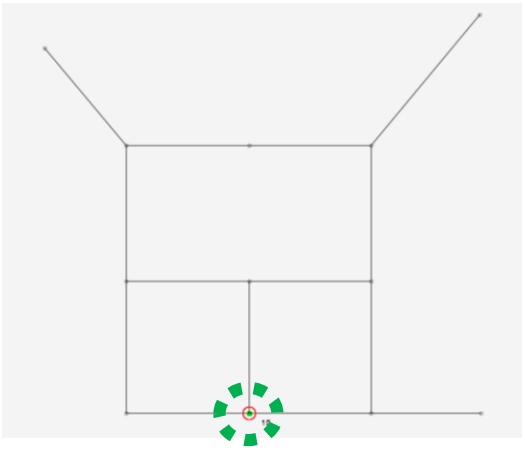
Sensor

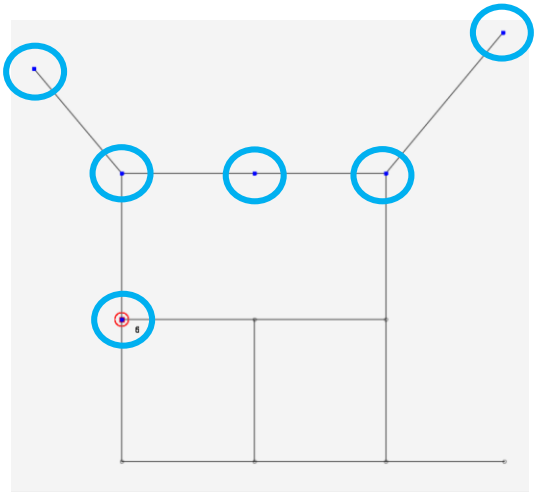
Probable origen de contaminación.

Origen de contaminación.

## Anexo 6 Análisis de inferencia en gessler1985.inp con dos sensores.







Leyenda:

Sensor

Probable origen de contaminación.

Origen de contaminación.

## Anexo 7 Análisis de inferencia en Exnet.inp con tres sensores.