Universidad de las Ciencias Informáticas

Facultad 5

SISTEMA DE SOPORTE DE DECISIÓN PARA ANÁLISIS DE OBSERVABILIDAD.

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

Autor: Alexis Rodríguez Paret.

Tutor(es): Ing. Yuremis Mengana Claro.

Ing. Arian Antonio Nuñez.

Ciudad de La Habana
Junio de 2011

Declaración de autoría.

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

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

_____
Firma del Autor: Alexis Rodríguez Paret

_____ _____
Firma del Tutor: Ing. Arian Antonio Nuñez Firma del Tutor: Ing. Yuremis Mengana Claro

Agradecimientos.

A la Revolución por permitir la posibilidad de estudiar a todos por igual.

A mi mamá:

Por la educación que me ha dado y siempre estar cuando la necesito aunque estemos lejos.

A mi papá:

Que aunque hace mucho no estamos cerca siempre me ha apoyado y confiado en mí.

A mi tía Esperanza:

Por ser la que lleva la paz a esta familia.

A mi Abuela Cacha:

Por haber sido la formadora de tan buena madre y tías.

A mi tía Alina:

Por siempre querer sacarnos adelante, y esforzarse en la formación de todos.

A mi tía Mairena:

Que aunque aparente ser molesta, y gruñona se preocupa por todos.

A mi abuela Nery:

Que siempre me ha adorado, y ayudado en todo lo que me ha hecho falta.

A Pancho:

Que aunque ya no está dejó parte importante de él todos.

A mi tío Borys:

Por ser mi ayuda y soporte todo este tiempo en la ausencia de mi padre.

A mi hermana Liset:

Por siempre haber sido la felicidad de la casa.

A mis primos de la loma:

Por ser el ejemplo que me impulso siempre a esforzarme en los estudios.

A René:

Por ser ese amigo de toda la vida.

A Juan Carlos:

Quien incondicionalmente ha sido mi amigo de la universidad.

A mi novia Irenna:

Con la que comparto todo y cuanto tengo.

A Marquito, Damir, Reinier los amigos de la loma.

A Yaniel, Antonio, Evelio, Alián, Dayanis, Yoana, Libeidys, Liusba, Liudmila, Yurisel, Dunia, Luis Mariano por compartir juntos parte de la universidad.

Dedicatoria.

A mi familia que siempre me ha apoyado, en especial a mi madre.

Alexis.

Resumen.

El análisis de observabilidad es un proceso fundamental en el área del diseño de implementación de plantas industriales. El proceso que se sigue para realizar el análisis de observabilidad consta de dos partes fundamentales, la creación de un modelo matemático de la planta, y ejecución de algoritmo de análisis de observabilidad.

La creación del modelo es el proceso que se encarga de crear ecuaciones matemáticas algebraicas que representan los balances de masa de todos los componentes, así como de las corrientes que los unen.

El análisis de observabilidad por su parte es el encargado de clasificar las variables no medidas del proceso en variables observables y no observables las cuales pueden obtenerse a partir de las variables medidas usando ecuaciones de balance.

Por la importancia que tiene el diseño de implementación en el monitoreo de procesos se desarrolló un sistema de soporte de decisión para el análisis de observabilidad capaz de asistir al ingeniero de procesos en la ardua tarea de tomar decisiones respecto a la configuración de instrumentación.

Palabras clave: Análisis de observabilidad, balance de masa, componentes, configuración de instrumentación, modelo matemático, monitoreo de procesos, variables no medidas.

Índice.

Índice de Figuras.

Índice de Tablas.

Introducción.

En la actualidad la industria petrolera representa un sector importante tanto en la economía de los países desarrollados y de la economía global como de la economía y desarrollo de países emergentes (Grossman, 2003). Debido a esto, se ha hecho un esfuerzo considerable para mejorar el diseño y la operación de las plantas tal que los procesos operen en forma segura y eficiente, se asegure la mejor calidad de los productos, se mantenga la viabilidad económica y la competitividad del negocio, y se reduzca el impacto ambiental de sus actividades.

Actualmente para la industria, la productividad es un factor crítico, solo minutos de paro, se puede traducir en grandes pérdidas. Es por esto que hoy en día han tenido un auge considerable los sistemas de control y adquisición de datos SCADA (Supervisory Control and Data Adquisition), dado que estas aplicaciones de software, especialmente diseñadas para funcionar sobre ordenadores en el control de producción, proporcionan comunicación y control de los dispositivos de campo (controladores autónomos y autómatas programables).

El concepto de SCADA se aplica entonces en todo proceso que exige un control de la calidad, producción y optimización de servicio. Petróleos de Venezuela Sociedad Anónima (PDVSA), es la corporación estatal de la República Bolivariana de Venezuela a cargo de la exploración, transporte y comercialización de los hidrocarburos. Esta corporación empleaba para el control de sus instalaciones varios sistemas SCADA suministrados por compañías privadas encargadas de brindar la seguridad y eficiencia operacional de los sistemas automatizados.

La Universidad de las Ciencias Informáticas (UCI) específicamente el Centro de Informática Industrial (CEDIN) en conjunto con Venezuela desarrolla el sistema SCADA Guardián del ALBA. Con la culminación de los primeros trabajos pactados y los avances generales alcanzados, el desarrollo del sistema SCADA Guardián del ALBA ha llegado a un punto en que resulta conveniente agregar funcionalidades que permitan que el producto se aproxime al estado de los sistemas SCADA que lideran el desarrollo de aplicaciones destinadas al control de procesos.

La efectividad de estos sistemas depende en gran medida de la cantidad de valores del proceso que manejen siendo así directamente proporcional su relación.

Muchas variables que son de gran interés en ocasiones no son medidas por la falta de un buen análisis de instrumentación el cual consiste en definir la cantidad, tipo y

ubicación de los sensores requeridos para lograr el conocimiento suficiente del estado real de la planta.

El sistema SCADA Guardián del ALBA está desplegado sobre plantas petroquímicas de proceso continuo en las que se mide un conjunto de variables, y otras no se miden por razones económicas o tecnológicas, provocando esto que no se tenga un conocimiento completo y confiable del funcionamiento de la planta, por lo que el SCADA necesita un sistema que permita obtener una mejor configuración de la instrumentación del proceso en las plantas, resultando esta la problemática. Por lo que surge entonces el problema científico a resolver ¿Cómo obtener un conocimiento más completo y confiable del funcionamiento de las plantas?

El análisis de observabilidad es un problema de especial interés dentro del área de monitoreo de procesos, dicho análisis se realiza a través de una representación de estado estacionario de la planta el cual puede ser modelado matemáticamente mediante un sistema de ecuaciones algebraicas. El análisis de observabilidad busca determinar cuáles variables no medidas pueden ser calculadas a partir de las mediciones y las ecuaciones del modelo, por lo que el **objeto de estudio** lo constituye: El análisis de observabilidad para el monitoreo de procesos, teniendo como **objetivo general**: Desarrollar un sistema de soporte de decisión (Decision Support System, DSS) para el análisis de observabilidad, y el **campo de acción**: El análisis de observabilidad para el SCADA Guardián del ALBA.

Para dar respuesta efectiva a los retos antes planteados, el objetivo general se desglosa en una serie de tareas de investigación:

- Elaborar los fundamentos teóricos sobre sistemas de soporte de decisión para análisis de observación.
- Elaborar solución propuesta.
- Analizar, diseñar y desarrollar un sistema de soporte de decisión para análisis de observación.

Para ello:

En el capítulo 1, se hace un estudio sobre el análisis de observabilidad en la monitorización de procesos, dándole al lector una panorámica sobre algoritmos de observabilidad para la clasificación de variables basados en teoría de grafos, además de la aplicación de los mismos en un sistema de soporte de decisión.

En el capítulo 2, se hace un estudio de las principales tecnologías usadas en este trabajo así como una descripción de la solución propuesta.

En el capítulo 3, se presentan diagramas de clases del análisis y el diseño, diagramas de secuencia, las descripciones de las clases del diseño, así como la implementación del sistema basado en los resultados del análisis y el diseño elaborado.

Anexo A, Imágenes de la aplicación.

Se muestran imágenes de la interfaz gráfica de la aplicación referentes a algunas de las etapas del proceso de modelación y resultados de la clasificación de variables.

Como resultado de este trabajo se pretende obtener un sistema de soporte de decisión para análisis de observación mediante el cual se logre obtener una mejor instrumentación de las plantas.

Capítulo 1. Fundamentación Teórica.

En la actualidad los grafos son muy utilizados en herramientas de modelado, dado que básicamente en muchas ocasiones son una abstracción útil de muchas situaciones del mundo real, especialmente en problemas de interconexión entre distintos objetos; técnicas sobre grafos como la búsqueda en profundidad (DFS) conocida en inglés como Depth-First-Search, constituyen la base de varios métodos de análisis de observabilidad tratados en esta tesis, por lo que la teoría de grafos constituye una base rigurosa a partir de la cual se fundamentan las técnicas y metodologías usadas en este trabajo.

El análisis de instrumentación se realiza para definir la cantidad, tipo y ubicación de sensores necesarios para obtener un conocimiento más completo del estado de la planta en cualquier instante de tiempo, lo cual permitiría detectar la presencia de instrumentos mal ubicados o innecesarios. Los datos de planta se pueden examinar empleando modelos matemáticos de estado estacionario que representen adecuadamente el funcionamiento de la misma, dichos modelos son conformados mediante ecuaciones algebraicas **E** representando los balances, relaciones termodinámicas y correlaciones experimentales, siendo a su vez los modelos no lineales los más rigurosos dado que estos involucran balances de masa y energía, relaciones termodinámicas para estimar densidad, entalpias y constantes de equilibrio.

En el monitoreo de procesos, los algoritmos de clasificación de variables no medidas también conocido como análisis de observabilidad son una herramienta fundamental, haciendo posible definir si la instrumentación de una planta es suficiente para conocer todas las variables de interés, las cuales pueden ser clasificadas según su factibilidad de cálculo, (Romagnoli y Sánchez, 1999):

1. **Variables Redundantes:** variables medidas que pueden ser calculadas a partir de los balances del resto de las variables medidas.
2. **Variables No Redundantes:** variables medidas que no pueden computarse a partir de los balances del resto de las variables medidas.
3. **Variables Observables:** variables no medidas que pueden obtenerse a partir de las variables medidas usando las ecuaciones de balance.
4. **Variables No Observables:** variables no medidas que no pueden calcularse a partir de las variables medidas mediante las ecuaciones de balance.

Por otra parte las ecuaciones **E** pueden clasificarse en tres grupos:

1. **Ecuaciones Asignadas:** aquellas que se emplean para despejar variables observables.
2. **Ecuaciones Redundantes:** son aquellas cuyas variables son todas observables o medidas, y que no se emplean para despejar variables observables.
3. **Ecuaciones No Asignadas:** son aquellas que contienen al menos una variable no observable y no pueden emplearse para despejar variables observables.

Una buena clasificación de variables no medidas mediante un análisis de observabilidad que logra determinar el mayor número de variables observables permite reducir la cantidad de sensores ubicados, resultando factible económicamente en costos de inversión evitando así mediciones innecesarias.

Un sistema de soporte de decisión para el análisis de observabilidad que permita facilitar al ingeniero de procesos la creación del diseño de instrumentación, dándole la capacidad de generar reportes, organizar, analizar y visualizar la información brindaría la posibilidad de tomar mejores decisiones de diseño, pudiendo así observar todos los factores que intervienen en la instrumentación de la planta.

1.1. Algoritmos de observabilidad

Un problema de especial interés dentro del área de monitoreo es el análisis de observabilidad siendo este un estudio que se realiza a partir de una representación de estado estacionario de la planta tal que nada cambie en el tiempo, pudiendo este entonces ser modelado mediante un sistema de ecuaciones no lineales, buscando obtener cuales variables no medidas del proceso pueden ser calculadas usando las mediciones y las ecuaciones del modelo.

Una vez elegido el modelo para representar un proceso los algoritmos de clasificación de variables permiten establecer si la instrumentación existente en una planta es suficiente para conocer todas las variables de interés y establecer la ubicación de los sensores, permitiendo obtener así una mejor configuración de instrumentación del proceso redundando este en múltiples beneficios en términos de calidad, medio ambiente, ahorrando simultáneamente dinero y recursos.

Existe gran cantidad de metodologías para la clasificación de variables, pero dada la complejidad de los procesos en las plantas actuales cuando se desea brindar una representación realista del problema es necesario usar modelos matemáticos que requieran de ecuaciones fuertemente no lineales.

Los algoritmos tratados en esta tesis se basan en efectuar reordenamientos estructurales de la matriz de ocurrencia asociada al sistema de ecuaciones que modela la planta.

1.1.1. Método Directo.

El método directo [Anexo A5] para análisis de observabilidad es una técnica basada en descomposición de grafos, este algoritmo consta de seis etapas como se muestra en la Figura 1.

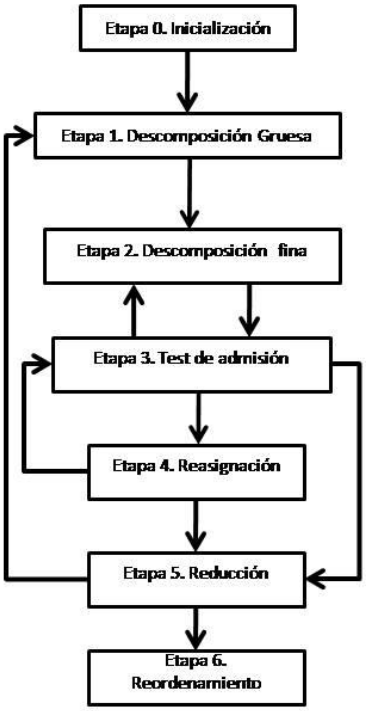


Figura 1 Esquema algorítmico del método directo.

En la primera etapa (Inicialización) se construye el bigrafo **G** asociado a la submatriz de ocurrencia **N.** Después se efectúa la descomposición gruesa de **G** mediante la búsqueda de un pareamiento maximal de **G**. Ya en la segunda etapa se efectúa un

segundo nivel de particionado donde se determinan los subconjuntos de asignación de **N**, para la tercera etapa cada uno de estos subconjuntos es testeado, y cuando uno de estos bloques resulta no admisible se efectúa una reasignación (etapa cuatro) este proceso trata de sustituir una de las filas del subconjunto rechazado por alguna fila correspondiente a una ecuación redundante. Si la reasignación es exitosa, se regresa al paso dos en otro caso se pasa al paso cinco de reducción, en esta fase, se eliminan de **G** todos los nodos asociados a las filas y columnas de los bloques que superaron el test de admisión. Luego, con el bigrafo reducido, se regresa a la etapa uno. Cuando ya no se detectan nuevos subconjuntos de asignación, el algoritmo pasa a la etapa seis donde se produce el reordenamiento y termina el procedimiento.

Este método resulta muy eficiente para la clasificación de variables para casos de plantas industriales de gran envergadura [14].

1.1.2. CDGH.

CDGH (Cycle Detection on Hypergraphs) es un algoritmo basado en hipergrafos, que efectúa la clasificación de variables no medidas para problemas de instrumentación, donde la calidad de las particiones logradas por este dependen en gran medida del diseño de las reglas de heurística.

1.1.3. GS-FLCN.

El método GS-FLCN [Anexo A1] conocido en inglés como ***Global Strategy with First Least-Connected Node*** consiste básicamente en descomponer la matriz de ocurrencia **M** asociada al modelo del proceso mediante una búsqueda incremental por tamaño para encontrar el máximo número de subconjuntos de asignación de tamaño mínimo. Las filas de la matriz **M** corresponden a las ecuaciones del modelo de la planta y sus columnas están asociadas a las variables no medidas. Un subconjunto de asignación es un bloque asociado a un subsistema de ecuaciones algebraicas cuadrado que admite solución numérica. En la Figura 2 se muestra un esquema algorítmico de GS-FLCN. El objetivo final del análisis de observabilidad consiste en permutar **M** para lograr el patrón esquematizado en la Figura 3.

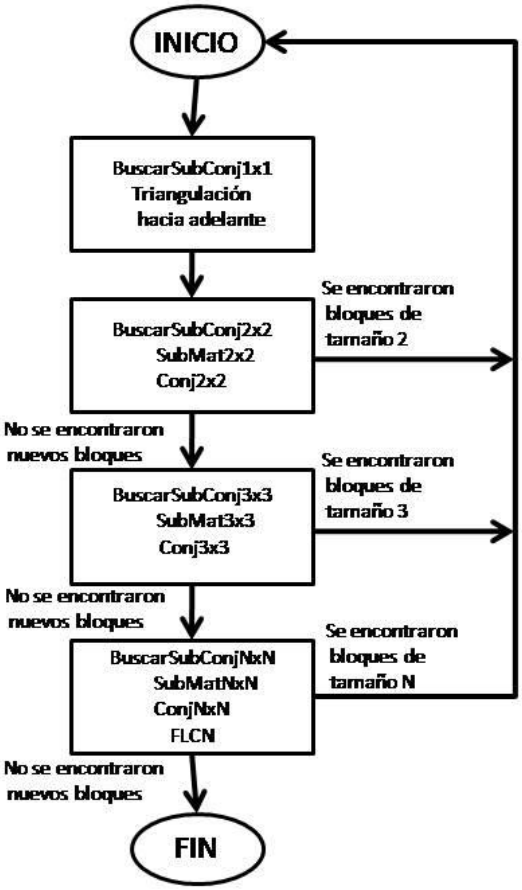


Figura 2 Esquema algorítmico de GS-FLCN

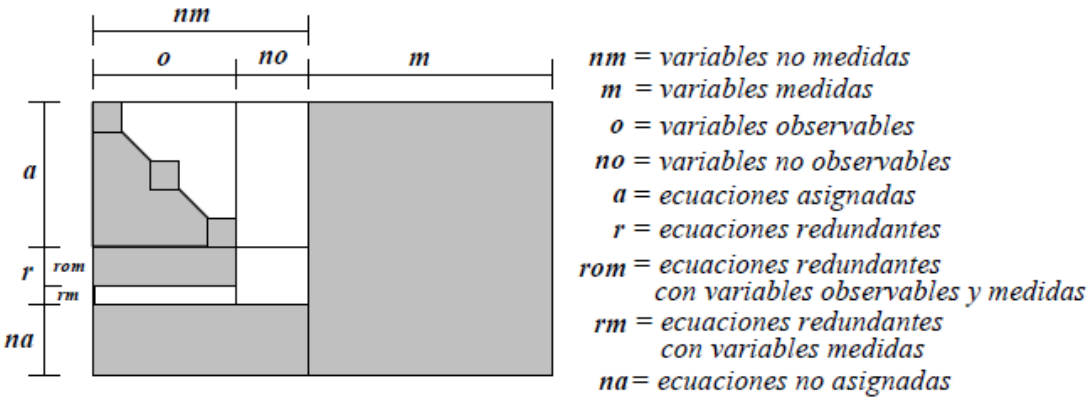


Figura 3 Matriz de ocurrencia reordenada.

Primeramente se emplea el método clásico de triangulación hacia delante para detectar variables calculables por despeje directo a partir de variables medidas (procedimiento *BuscarSubConj1x1*) [Anexo A1]. Luego se utilizan las rutinas

BuscarSubConj2x2 [Anexo A2] y *BuscarSubConj3x3* [Anexo A3] para detectar subconjuntos de tamaño 2 y 3 respectivamente. Finalmente la rutina *BuscarSubConjNxN* [Anexo A4] se encarga de buscar los bloques de tamaño 4 en adelante, donde cada uno de estas rutinas hace uso de procedimientos como *SubMatXxX* y *ConjXxX*, donde el primero se encarga de construir la submatriz utilizada por *ConjXxX* para realizar la búsqueda.

El procedimiento BuscarSubConjNxN es el más costoso con respecto al tiempo de computo pues se encarga de detectar los subconjuntos de mayor tamaño; donde cada subconjunto de asignación de tamaño *m* puede asociarse a un camino de longitud *m* - 1 en **G**, donde los nodos corresponden a las variables observables. La rutina *FLCN* utiliza un algoritmo de búsqueda DFS para encontrar todos los caminos de longitud *m*-1 sobre el grafo no dirigido **G** correspondiente a la matriz de ocurrencia.

Este algoritmo exhibe buen desempeño en cuanto a robustez dado que encuentra gran cantidad de subconjuntos de asignación, además de tener la capacidad para manejar sistemas altamente no lineales. Un aspecto de interés de GS-FLCN es que la descomposición estructural obtenida por el método conduce a una configuración con una mínima cantidad de instrumentos, permitiendo a su vez una fácil visualización de los lugares más convenientes para adicionar o remover sensores. Conduciendo esto a importantes ahorros en inversiones y costos operativos.

GS-FLCN resulta una estrategia confiable, robusta y eficiente para la clasificación de variables en problemas de instrumentación de plantas de procesos, el mismo posee un amplio rango de aplicación gracias a su enfoque estructural, pudiendo ser aplicada en modelos matemáticos fuertemente no lineales [14].

1.2. **Grafos. Algunas definiciones.**

Un grafo no dirigido **G** puede representarse simbólicamente como **G** = (**N**, **E**), donde **N** es un conjunto de nodos y **E** es una colección (no necesariamente un conjunto) de pares no ordenados de nodos, que se denominan aristas. Si *u* y *v* son nodos de **G** y existe un par no ordenado *a* = (*u*, *v*) en **E**, se dice que a une *u* y *v*, ó que existe una arista a entre *u* y *v*. En este caso, también se dice que *u* y *v* son incidentes en *a*, ó que *a* tiene incidencia en *u* y *v*. Por otra parte, diremos que *u* es adyacente a *v* y *v* es adyacente a *u*. Un grafo **G** = (**N**, **E**) se denomina grafo dirigido o digrafo cuando las aristas *e* = (*u*, *v*) de **E** son dirigidas. En este caso se dice que la arista *e* parte de *u* y llega a *v*, ó que *v* es adyacente a *u*.

Cuando dos o más aristas conectan un mismo par de nodos, estas se denominan aristas paralelas. Una arista cuyos extremos coinciden se denomina bucle. Se dice que **G** es un grafo simple, si **G** no contiene aristas paralelas ni bucles. Además, un grafo simple **A** se dice un árbol si dados dos nodos cualesquiera *u* y *v* de **A**, existe un único camino entre *u* y *v*. Por otra parte, un grafo **S** = (**N***s*, **E***s*) es un subgrafo de **G** = (**N**, **E**) si **N***s* subconjunto de **N** y **E***s* subconjunto de **E**.

1.3. Sistema de soporte de decisión para observabilidad.

Los sistemas de soporte de decisión son paquetes de software que permitan facilitar la labor del profesional, dado que estos básicamente son sistemas basados en tecnologías computacionales coherentes que asisten al usuario en la toma de decisiones, estos sistemas se centran en mejorar la efectividad en la toma de decisiones complejas, donde la diversidad de aspectos a tener en cuenta y la forma en que estos se interrelacionan tornan muy difícil la labor en el momento de elegir entre distintas opciones igualmente factibles, en general las tareas abordadas por este tipo de software involucran el manejo de una gran cantidad de datos o caminos de decisión alternativos.

En el campo de la ingeniería de procesos las tendencias actuales conducen al desarrollo de paquetes computacionales capaces de analizar una planta de procesos químicos en su totalidad de forma conjunta.

Por lo que el análisis de observabilidad se beneficia en gran medida con el desarrollo de un sistema de soporte de decisión dado que la elección de los lugares para la ubicación de los instrumentos en una planta de procesos requiere del análisis sistemático de grandes cantidades de información heterogénea.

1.4. Paquete de software para análisis de observabilidad en diseño de instrumentación.

Este paquete de software [14] se concibió como un DSS para análisis de observabilidad usando como algoritmo base el Método Directo tratado en la sección 1.1.2, el diseño del mismo era altamente estático y no contemplaba un escenario cambiante relacionado con el tamaño de los casos de estudio, la matriz de ocurrencia se implementaba mediante un arreglo estático con una capacidad máxima de ecuaciones y variables que podía contener, dadas estas características las consecuencias se veían reflejadas en que o bien existían casos de estudio que no

podían desarrollarse o se producía un desperdicio de espacio en memoria considerable, además de contar con una interfaz gráfica poco amigable para procesos tan complejos, razones por las cuales en 2006 dicho software fue llevado a un proceso de reingeniería [2] con el que se propuso trabajar con estructuras dinámicas lo cual le permitió el manejo transparente de arreglos dinámicos, acceso directo a elementos del arreglo y acceso secuencial a datos de la matriz; también fue modificada su interfaz gráfica anterior [Anexo B1] por una [Anexo B2] que permitiera la adición transparente de paquetes de software que implementan las distintas etapas del diseño de implementación, permitiéndole así al ingeniero de procesos abstraerse de las características del lenguaje de programación, o paradigma utilizado para la implementación del sistema y poder lograr un enfoque total en el diseño. En este sistema los modelos logrados a partir de la topología de una planta en la pantalla principal pueden ser visualizados tanto las ecuaciones, como las variables medidas y no medidas del sistema [Anexo B3], logrando entonces obtener con este nuevo software una mejor calidad con respecto a su antecesor.

1.5. Características de los sistemas SCADA.

El sistema más utilizado en la automatización de procesos industriales son los SCADA (Supervisory Control and Data Acquisition), estos sistemas se utilizan para controlar y monitorear dichos procesos, mediante dispositivos de campo (controladores autónomos, autómatas programables, etc.), de los cuales se recolecta información para mostrársela luego a un operador en forma amigable en un ordenador.

Los SCADA mejoran la eficacia del proceso de monitoreo y control proporcionando la información oportuna para poder tomar las decisiones operacionales apropiadas teniendo así un mejor manejo de las variables y los procesos involucrados en la producción.

La adquisición de los datos comienza en los dispositivos de campo, la información que estos generen luego es procesada por el sistema SCADA de manera que los operadores en el centro de control puedan supervisar todo el proceso.

1.5.1. Módulos de los Sistemas SCADA.

Los sistemas SCADA están compuestos por módulos o bloques de software que permiten las actividades de adquisición, supervisión y control.

Manejadores: Aseguran mediante una interfaz genérica la comunicación del sistema de supervisión y control con los distintos dispositivos que existen en el campo, ya sean autómatas, PLC, sensores inteligentes, etc.

Núcleo de Procesamiento de Datos: Representa el núcleo principal del procesamiento de los datos, es el encargado del procesamiento y análisis de la información recogida del campo a través de los manejadores. Una vez procesada esta información, es enviada al módulo que la requiera.

Base de Datos Históricos: Aquí se implementa el mecanismo encargado del almacenamiento de la información recibida desde el campo, así como la sucesión de alarmas y eventos generados. La información almacenada es utilizada por varias aplicaciones del sistema.

Middleware: El Middleware es la capa de software, que se encarga de la comunicación entre los diferentes procesos distribuidos de medio y alto nivel, que forman parte del sistema SCADA y el principal elemento que caracteriza su complejidad es la gestión de las comunicaciones.

Ambiente de Configuración: Esta aplicación permite configurar varios procesos o partes de ellos, aquí se definen y gestionan las variables, la configuración de los manejadores, los comandos, las alarmas y variadas opciones adicionales. Este ambiente funciona como una aplicación de diseño tradicional, con la peculiaridad que los sinópticos se confeccionan a partir de objetos y primitivas básicas predefinidas, que se pueden agrupar, combinar, transformar, importar y exportar entre otras.

Ambiente de Ejecución (Run-time): Aplicación encargada de la visualización, la cual permite representar gráficamente de manera fidedigna los procesos operacionales con los que el usuario interactúa. A través de esta aplicación se pueden visualizar las condiciones operacionales, valores de variables, navegar entre despliegues, visualizar alarmas, enviar comandos, entre otros.

1.5.1.1. Módulo de Aplicaciones.

Con el desarrollo tecnológico alcanzado por el hombre los Sistemas de Control y Adquisición de Datos fueron usando con mayor frecuencia funcionalidades de optimización y algoritmos de control avanzado creados por los usuarios para lograr mayor eficiencia y complementar las técnicas de control utilizadas a nivel de campo.

Por lo que debido a esto se desarrolló un sistema especialmente concebido esencialmente para ser integrado al SCADA Guardián del ALBA, con el objetivo de adicionar un grupo de funcionalidades necesarias para el control avanzado, agregando así un importante valor a la supervisión, el control de procesos y la obtención de mayores niveles de eficiencia y eficacia operacional.

El módulo de aplicaciones (App) permite el control y la ejecución de algoritmos, admitiendo una planificación basada en la configuración y ejecución de tareas periódicas, no periódicas, síncronas y asíncronas. La solución establece un esquema de interacción que permite, a los algoritmos creados por los usuarios, interactuar con todos los servicios del SCADA y aplicaciones externas que dan soporte a la toma de decisiones así como a la simulación de procesos.

Una de las funcionalidades del módulo de aplicaciones es que este ofrece al usuario un editor de algoritmos, con el cual se pueden editar programas en un lenguaje de programación de alto nivel, orientado a objetos, que permita la utilización de bibliotecas matemáticas entre otras, el cual en esta primera versión en la que se encuentra se realizó usando una interfaz de programación de aplicaciones (application programming interface, API) de C++ integrada al lenguaje de programación Python.

1.6. Metodología de Ingeniería de Software.

Para realizar el análisis y diseño de esta tesis se uso RUP (Proceso Unificado de Desarrollo), por que acumula muchos años en desarrollo de software donde la experiencia es un factor que la hace mas completa y confiable, la misma está rigurosamente probada a nivel mundial con innumerables sistemas de software, las características mas importantes que se tuvieron en cuenta para su selección fueron las siguientes:

- **Guiado por casos de uso**: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, constituye la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo del sistema.
- **Centrado en la arquitectura**: La arquitectura muestra la visión común del sistema completo.
- **Iterativo e incremental**: RUP divide el proyecto en fases de desarrollo, propone además que cada una de ellas se desarrolle en iteraciones, las

cuales aportan un incremento en el proceso de desarrollo y terminan con el cumplimiento del punto control trazado en la fase.

1.7. **Python + Qt.**

Python es un lenguaje interpretado o de script independiente de la plataforma y orientado a objetos preparado para realizar cualquier tipo de programa, por lo que en los últimos años se ha hecho muy popular por varias razones como:

- La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el proprio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.
- La sencillez y velocidad con la que se crean los programas. Un programa en Python puede tener de 3 a 5 veces menos líneas de código que su equivalente en **Java** o **C**.
- La cantidad de plataformas en las que podemos desarrollar, como Unix, Windows, OS/2, Mac, Amiga y otros.
- Python es gratuito, incluso para propósitos empresariales.

Python también puede utilizarse como un lenguaje de extensión para módulos y aplicaciones que necesitan una interfaz programable.

Qt es una biblioteca multiplataforma, creada para desarrollar interfaces graficas de usuario, esta es desarrollada con el lenguaje de programación **C++**, de forma nativa pero existen módulos para otros leguajes de programación como son C, Python (**PyQt**), Java (**Qt Jambi**) entre otros.

Qt es una librería totalmente orientada a objetos, es por ello que las **API** (Application Programming Interface) cuenta con diferentes métodos, soportando el uso de diferentes motores de Bases de Datos y el uso de archivos **XML**, además de otras estructuras de datos tradicionales[8].

1.8. **IDE Eclipse.**

El eclipse es un Entorno Integrado de Desarrollo multiplataforma libre para crear aplicaciones clientes de cualquier tipo. Con este IDE se han desarrollado importantes aplicaciones como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

El IDE de Eclipse emplea módulos para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no, este mecanismo permite que el entorno de desarrollo soporte otros lenguajes además de **Java**, como por ejemplo un módulo para dar soporte a Python.

1.9. **Conclusiones del capítulo.**

En este capítulo se realizó un esbozo de los principales algoritmos de observabilidad propuestos por Ponzoni [14].También se abordó sobre el tema de los sistemas de soporte de decisión para el análisis de observabilidad en el monitoreo de procesos. Además se realizó una breve descripción de los sistemas SCADA haciendo especial énfasis en el módulo de Aplicaciones, así como también se presentan las posibles tecnologías a utilizar en el desarrollo de la propuesta de solución. A partir de estos puntos se comenzará el desarrollo de la propuesta de solución.

2. Solución Propuesta.

En el presente capítulo se adquiere una visión práctica del sistema que se pretende desarrollar. En el mismo se expondrán las reglas del negocio, así como los requisitos funcionales y no funcionales que regirán el desarrollo de la solución al problema. Partiendo de esto se determinarán los casos de uso a realizar. Además se describirán los procesos de las principales funcionalidades del sistema.

2.1. Comparación entre algoritmos.

Los algoritmos mencionados en el capítulo anterior fueron comparados en varios casos de estudio [14] con el objeto de analizar su desempeño; en la tabla 1 se muestran los resultados de la comparación realizada entre el CDHG y GS-FLCN para el caso de una planta industrial, en esta prueba el algoritmo GS-FLCN fue ejecutado sin factores de ramificación.

Área	**Caso 1**		**Caso 2**	
Dimensión	**772x529**		**333x258**	
Tamaño de bloques	**GS-FLCN**	**CDHG**	**GS-FLCN**	**CDHG**
1	407	397	200	181
2	2	3	6	3
3	6	6	6	1
4	-	1	2	4
5	-	-	1	4
6	-	2	-	2
8	1	-	-	-
10	-	-	1	-
13	-	-	-	1
pva	82.6 %	82.6 %	98.4 %	91.5 %
Tiempo de ejecución (seg)	9450	68	1473	33

Tabla 1 Comparación entre algoritmos CDHG y GS-FLCN

La primera columna de la tabla indica el tamaño de los subconjuntos de asignación y las últimas dos filas muestran el porcentaje de variables asignadas (***pva***) y los tiempos de ejecución, respectivamente. En estos dos casos de estudio se pudo apreciar que el

CDHG demostró mejor comportamiento en cuanto a tiempo de ejecución aunque el ***pva*** disminuyó ligeramente y esto se debe a la buena calidad de las reglas de heurística seleccionadas para guiar la búsqueda, quedando evidente que este método depende en gran medida de las reglas que le sean definidas. En definitiva CDHG puede obtener la clasificación de variables en menores tiempos de cómputo que GS-FLCN, pero GS-FLCN es significativamente más robusto.

En casos de estudio realizados en las que el modelo matemático está formado por 104 ecuaciones algebraicas no lineales con 25 variables medidas y 85 no medidas, el algoritmo GS-FLCN logra mayor robustez que el método directo dado que detecta mayor cantidad de bloques de mínimo tamaño incluso cuando ambos clasifican como observables las mismas variables; demostrando así que en modelos de pequeño tamaño GS-FLCN logra una descomposición de granularidad igual o más fina que el método directo, los resultados son mostrados en la tabla 2.

Dimensión	**104x100**	
Tamaño de bloques	**GS-FLCN**	**Método directo**
1	55	44
2	1	1
5	-	1
6	1	-
12	-	1
pva	74%	74%
Tiempo de ejecución (en minutos)	1:53	0:03

Tabla 2 Comparación entre algoritmos GS-FLCN y Método directo

GS-FLCN fue probado sobre modelos de gran envergadura donde al mismo se le aplicó factores de ramificación para lograr tiempos de cómputo razonables, en los que demostró poca eficiencia en cuanto a la cantidad de subconjuntos detectados, siendo la principal razón el hecho de que se le acotara significativamente la capacidad de exploración.

De la comparación realizada entre estos dos últimos métodos se pudo obtener que en plantas de industriales de gran dimensión en donde los subconjuntos de asignación son grandes el método directo posee un desempeño fuertemente superior respecto a GS-FLCN, por lo que es recomendable que en casos de gran envergadura usar el método directo o bien particionar la planta en secciones apropiadas, y luego realizar el

análisis de observabilidad de cada zona antes modelada usando GS-FLCN, siendo esta muy atractiva para el desarrollo de estudios de tipo local, centrado en los equipos más críticos del proceso industrial.

2.2. **Consideraciones técnicas generales.**

El campo del análisis de observabilidad obtiene un gran beneficio con el desarrollo de un DSS, dado que en casos como estos donde el usuario tiene que intervenir en la toma de las decisiones claves durante el diseño de la instrumentación, la aplicación de un sistema de expertos no resulta factible, teniendo como principal razón que la obtención en forma completamente automática de los resultados finales podría conducir a soluciones poco confiables, lo cual viene dado por que muchas decisiones involucradas en el análisis son subjetivas y muy específicas para cada caso.

Por su parte resulta muy conveniente desarrollar un DSS que permita al ingeniero de procesos interactuar con el software y jugar un rol más activo durante el diseño de instrumentación.

Por lo que para darle cumplimiento al objetivo de este trabajo, se pretende desarrollar un DSS para el Análisis de Observabilidad desarrollado en PyQt(binding de la biblioteca grafica Qt para el lenguaje de programación Python)logrando así que el mismo sea independiente de la plataforma en que sea ejecutado, dicho sistema funcionará como una tarea del módulo de aplicaciones, pudiendo así en futuras iteraciones facilitar la integración de funcionalidades mas ambiciosas dada la facilidad que brinda App para la comunicación con otros módulos del SCADA; este sistema tendrá como funciones, gestionar la topología de la planta, así como las ecuaciones, variables y sensores correspondientes al modelo, para lo cual se obtuvieron un conjunto de características que se describen a continuación.

2.3. **Modelo Dominio.**

A continuación se representa un acercamiento a la solución propuesta, donde se modelan los principales conceptos con los que se trabajarán en el desarrollo de la solución, así como las relaciones existentes entre ellos.

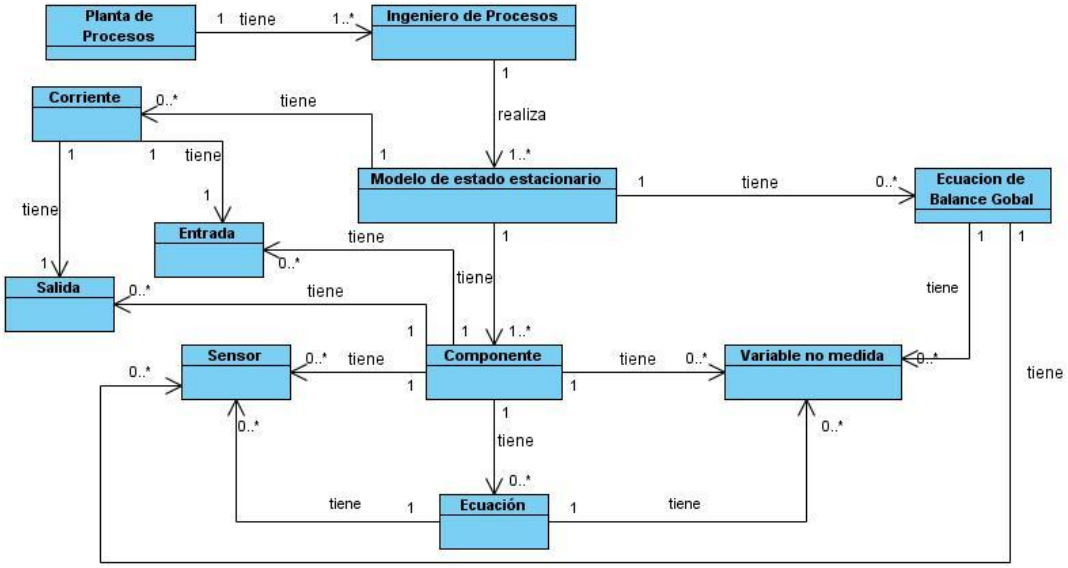


Figura 4 Modelo de dominio de la aplicación.

2.3.1. Descripción del modelo de dominio.

Planta de Procesos, planta industrial.

Ingeniero de Procesos es el encargado del diseño de la planta y creación del diseño de instrumentación.

El modelo de estado estacionario es el conjunto de ecuaciones matemáticas algebraicas que modelan una planta en una combinación de condiciones tal que nada cambia en el tiempo.

Ecuaciones de Balance Global son aquellas que involucran variables de diferentes componentes del modelo de la planta.

Componente se refiere a los equipos existentes en la planta ya sean calderas, intercambiadores de calor, separadores, mezcladores entre otros.

Las ecuaciones por componente son aquellas que involucran variables del mismo componente.

Variable no medida se refiere a aquella para la cual no se tiene un sensor para obtener su valor, como podrían ser temperatura, presión entre otras.

Sensor es aquel instrumento capaz de medir magnitudes físicas o químicas, llamadas variables de instrumentación, como podrían ser temperatura, presión, humedad entre otras.

Salida se refiere al punto de salida de cierta materia de un componente.

Entrada se refiere al punto de entrada de cierta materia a un componente.

Corriente, tuberías que esta caracterizada por un caudal de materia, composición de mezcla circulante, temperatura y presión, la cual por lo general posee una o varias entradas y salidas.

2.4. Captura de requisitos.

Los requisitos constituyen capacidades o condiciones que el sistema debe cumplir. A continuación se exponen los requisitos funcionales por los que se regirá el sistema y los no funcionales que exponen las características de la aplicación.

2.4.1. Requisitos funcionales.

R1. Gestionar Modelo.

R1.1. Salvar modelo.

R1.2. Cargar modelo.

R1.3. Gestionar componente.

R1.3.1 Insertar componente.

R1.3.2 Eliminar componente.

R1.3.3 Salvar componente.

R1.3.4 Cargar componente.

R1.4. Gestionar sensor.

R1.4.1 Insertar sensor.

R1.4.2 Eliminar sensor.

R1.5. Gestionar variable no medida.

R1.5.1 Insertar variable no medida.

R1.5.2 Eliminar variable no medida.

R1.6. Gestionar corriente.

R1.6.1 Insertar corriente.

R1.6.2 Eliminar corriente.

R1.7. Gestionar ecuación.

R1.7.1 Adicionar ecuación.

R1.7.2 Eliminar Ecuación.

R2. Ejecutar análisis de observabilidad.

El sistema posibilitará:

R2.1. Clasificar variables no medidas en observables.

R2.2. Mostrar matriz de ocurrencia.

R2.3. Mostrar subconjuntos de asignación detectados.

2.4.2. Requisitos No Funcionales.

- **Usabilidad**: Estará dirigida a usuarios con conocimientos sobre diseño e instrumentación de plantas.
- **Diseño e implementación:** Se usará como lenguaje de programación Python y como biblioteca gráfica Qt, bajo el paradigma de programación orientado a objeto.
- **Soporte**: Multiplataforma.
- **Hardware**: Microprocesador superior a Intel Pentium 4 a 3.0 Ghz, memoria RAM superior a 512 Mb DDR2.
- **Legales**: Se regirá por la normal ISO 9000.

2.5. Modelo de casos de uso del sistema.

En esta sección se reconocen los posibles actores del sistema a desarrollar, y se definen los casos de uso del sistema. Además, se seleccionan los casos de uso correspondientes al primer ciclo de desarrollo para hacerles sus especificaciones textuales en formato expandido.

2.5.1. Actores del sistema.

Los actores de un sistema son agentes externos, roles que las personas o dispositivos juegan cuando interactúan con el software. En este caso particular quien hará uso del sistema será un ingeniero de procesos, que como actor del sistema será llamado Ingeniero.

Actores	Descripción
Ingeniero	Es quien interactúa con el sistema para ejecutar las funcionalidades brindadas por el sistema.

Tabla 3 Descripción de los Actores del Sistema.

2.5.2. Diagrama de casos de uso del sistema.

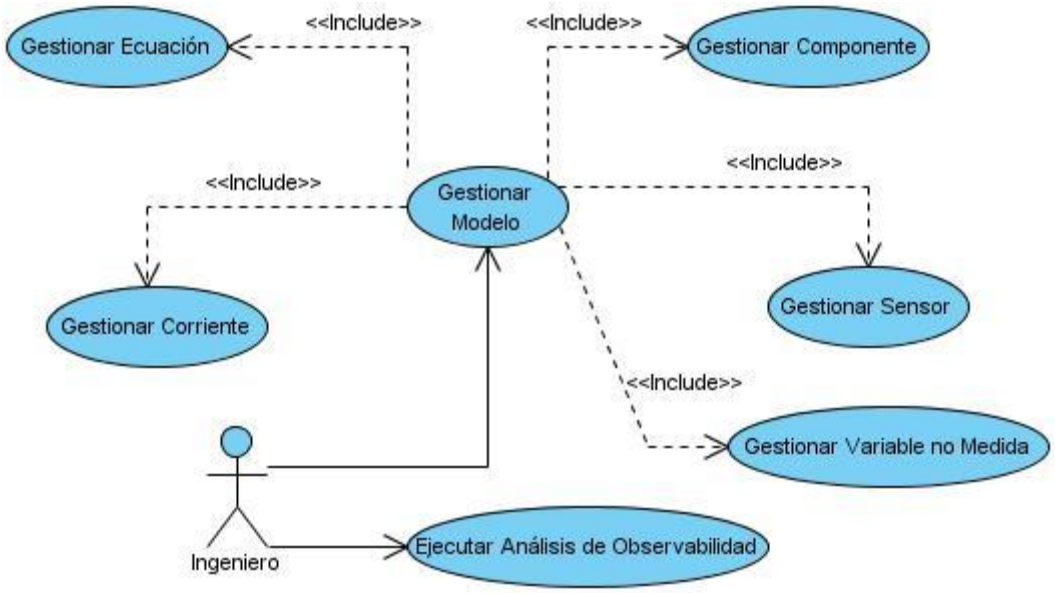


Figura 5 Diagrama de casos de uso del sistema.

2.5.3. Descripción de casos de uso del sistema.

Caso de Uso:	**Gestionar componente**
Actores:	Ingeniero
Propósito	Su propósito es permitir al ingeniero crear, modificar, salvar y cargar componentes.
Resumen:	El caso de uso comienza cuando el ingeniero selecciona un componente que desee adicionar al modelo, este puede ser eliminado, salvado o cargado desde un fichero salvado con anterioridad.
Precondiciones:	
Referencias	R1.3.1, R1.3.2, R1.3.4, R1.3
Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. El actor puede seleccionar la Opción Insertar Componente, Eliminar Componente, Salvar Componente o Cargar Componente.	1.1. El sistema ejecuta alguna de las siguientes acciones: a) Si el actor desea insertar un componente consultar sección Insertar Componente. b) Si el actor desea eliminar un componente consultar sección Eliminar Componente. c) Si el actor desea salvar un componente consultar sección Salvar Componente. d) Si el actor desea cargar un componente consular sección Cargar Componente.
Pos condiciones	Se inserta un componente o se elimina un componente o se salva un componente o se carga un componente.

Tabla 4 Descripción del caso de uso Gestionar Componente.

Caso de Uso:	Insertar Componente
Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. Selecciona el componente que desea insertar.	1.1. La escena se activa en modo "Insertar Componente".
2. Selecciona la posición de la escena donde desea insertar el componente.	2.1. Se muestra en la escena el componente antes seleccionado. 2.2. La escena activa el modo "Mover componente"
Pos condiciones	Se inserta un componente en la escena.

Tabla 5 Descripción del caso de uso Insertar Componente.

Caso de Uso:	Eliminar Componente
Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. Selecciona el componente que desea eliminar.	
2. Selecciona la opción eliminar componente.	2.1. Elimina el componente de la escena.
Pos condiciones	Se elimina un componente en la escena.

Tabla 6 Descripción del caso de uso Eliminar Componente.

Caso de Uso:	Cargar Componente
Propósito	Añadir componentes ya existentes al sistema.
Prioridad	Secundario
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Negocio
1. Selecciona la opción cargar componente.	1.1. Muestra un cuadro de diálogo solicitando en fichero a cargar.
2. Selecciona el fichero que desea cargar.	2.1. Carga el fichero seleccionado por el actor. 2.2. Añade el componente cargado a la barra de “Nuevos Componentes
Pos condiciones	Se adiciona un nuevo componente al sistema.

Tabla 7 Descripción del caso de uso Cargar Componente.

Caso de Uso:	Salvar Componente
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. Selecciona el componente que desea salvar.	
2. Selecciona la opción Salvar Componente.	2.1. Muestra un cuadro de dialogo solicitando el nombre y la dirección donde desea guardar el componente seleccionado.
3. Inserta el nombre y selecciona la dirección donde desea salvar el componente.	3.1 El sistema guarda en un fichero el componente antes seleccionado.
Pos condiciones	Se salva en un fichero un componente.

Tabla 8 Descripción del caso de uso Salvar Componente.

Caso de Uso:	**Gestionar Sensor**
Actores:	Ingeniero
Propósito	Su propósito es permitir al ingeniero insertar, eliminar sensores a los componentes del modelo.
Resumen:	El caso de uso comienza cuando el ingeniero selecciona un componente al cual le desea adicionar o eliminar un sensor.
Precondiciones:	Debe haber sido insertado algún componente al modelo.
Referencias	R1.3, R1.4.1, R1.4.2
Prioridad	Crítica

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
2. El actor puede seleccionar la Opción Insertar Sensor, o Eliminar Sensor	2.1. El sistema ejecuta alguna de las siguientes acciones: e) Si el actor desea insertar un sensor consultar sección Insertar Sensor. f) Si el actor desea eliminar un sensor consultar sección Eliminar Sensor.
Pos condiciones	Se inserta un sensor o se elimina un sensor.

Tabla 9 Descripción del caso de uso Gestionar Sensor.

Caso de Uso:	Insertar Sensor	
Prioridad	Secundario	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Negocio**
1. Selecciona el componente al que le desea adicionar un sensor.		
2. Selecciona la opción Insertar Sensor.		2.1. Selecciona en la escena la posición donde desea ubicar el sensor. 2.2. Muestra un sensor en la posición seleccionada. 2.3. El sensor es adicionado a la lista de sensores del componente seleccionado.
Pos condiciones	Se adiciona un sensor al modelo.	

Tabla 10 Descripción del caso de uso Insertar Sensor.

Caso de Uso:	Eliminar Sensor	
Prioridad	Secundario	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Negocio**
1. Selecciona el sensor que desea eliminar.		

2. Selecciona la opción Eliminar Sensor.	2.1. Elimina todas las ecuaciones en las que se encontraba el sensor seleccionado. 2.2. Elimina el sensor del modelo.
Pos condiciones	Se elimina un sensor del modelo.

Tabla 11 Descripción del caso de uso Eliminar Sensor.

Caso de Uso:	**Gestionar Variable no Medida**
Actores:	Ingeniero
Propósito	Su propósito es permitir al ingeniero insertar, eliminar variables no medidas a los componentes del modelo.
Resumen:	El caso de uso comienza cuando el ingeniero selecciona un componente al cual le desea adicionar o eliminar una variable no medida.
Precondiciones:	Debe haber sido insertado algún componente al modelo.
Referencias	R1.3, R1.5.1, R1.5.2
Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
3. El actor puede seleccionar la Opción Insertar Variable no Medida, o Eliminar Variable no Medida.	3.1. El sistema ejecuta alguna de las siguientes acciones: a) Si el actor desea insertar una nueva variable no medida consultar sección Insertar Variable no Medida. b) Si el actor desea eliminar una variable no medida consultar sección Eliminar Variable no Medida.
Pos condiciones	Se inserta una nueva variable no medida o se elimina una variable no medida.

Tabla 12 Descripción del caso de uso Gestionar Variable no Medida.

Caso de Uso:	Insertar Variable no Medida
Prioridad	Secundario
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Negocio
1. Selecciona el componente al que le desea adicionar una se variable no medida.	
2. Selecciona la opción Insertar Variable no Medida.	2.1. Selecciona en la escena la posición donde desea ubicar la variable no medida. 2.2. Muestra una variable no medida en la posición seleccionada. 2.3. La variable no medida es adicionada a la lista de variables no medidas del componente seleccionado.
Pos condiciones	Se adiciona una variable no medida al modelo.

Tabla 13 Descripción del caso de uso Insertar Variable no Medida.

Caso de Uso:	Eliminar Variable no Medida
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. Selecciona la variable no medida que desea eliminar.	
2. Selecciona la opción Eliminar Variable no Medida.	2.1. Elimina todas las ecuaciones en las que se encontraba la variable no medida. 2.2. Elimina la variable no medida del modelo.
Pos condiciones	Se elimina una variable no medida del modelo.

Tabla 14 Descripción del caso de uso Eliminar Variable no Medida.

Caso de Uso:	**Gestionar Corriente**
Actores:	Ingeniero
Propósito	Su propósito es permitir al ingeniero insertar, eliminar corrientes entre las entradas y salidas de los componentes del modelo.
Resumen:	El caso de uso comienza cuando el ingeniero selecciona la opción insertar corriente, esta también puede ser eliminada.
Precondiciones:	Debe haber sido insertado como mínimo dos componentes al modelo.

Referencias	R1.3.1, R1.6.1, R1.6.2
Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
4. El actor puede seleccionar la Opción Insertar Corriente, o Eliminar Corriente.	4.1. El sistema ejecuta alguna de las siguientes acciones: c) Si el actor desea insertar una nueva corriente consultar sección Insertar Corriente. d) Si el actor desea eliminar corriente consultar sección Eliminar Corriente.
Pos condiciones	Se inserta una nueva corriente o se elimina una corriente.

Tabla 15 Descripción del caso de uso Gestionar Corriente.

Caso de Uso:	Insertar Corriente
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. Selecciona la opción Insertar Corriente.	
2. Selecciona una salida de un componente y una entrada de otro componente.	2.1. Crea una corriente entre la salida y entrada de los componentes respectivamente.
Pos condiciones	Se adiciona una corriente al modelo.

Tabla 16 Descripción del caso de uso Insertar Corriente.

Caso de Uso:	Eliminar Corriente
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. Selecciona la corriente que desea eliminar.	

2. Selecciona la opción Eliminar Corriente.	2.1. Elimina la corriente del modelo.
Pos condiciones	Se elimina una corriente del modelo.

Tabla 17 Descripción del caso de uso Eliminar Corriente.

Caso de Uso:	**Gestionar Modelo**
Actores:	Ingeniero
Propósito	Permitir salvar y cargar modelos previamente guardados en ficheros.
Resumen:	El caso de uso comienza cuando el ingeniero selecciona la opción salvar modelo, el mismo también puede ser cargado.
Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
5. El actor puede seleccionar la Opción Salvar Modelo, o Cargar Modelo.	5.1. El sistema ejecuta alguna de las siguientes acciones: e) Si el actor desea salvar un modelo consultar sección Salvar Modelo. f) Si el actor desea cargar un modelo sección Cargar Modelo.
Pos condiciones	Se salva un modelo o se carga un modelo.

Tabla 18 Descripción del caso de uso Gestionar Modelo.

Caso de Uso:	Cargar Modelo
Propósito	Cargar un modelo guardado en un fichero con anterioridad para el sistema.
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. Selecciona la opción cargar modelo.	1.1. Muestra un cuadro de diálogo solicitando en fichero a cargar.

2. Selecciona el fichero que desea cargar.	2.1. Carga el fichero seleccionado por el actor. 2.2. Muestra el modelo cargado en la escena
Pos condiciones	Se carga un modelo en la escena.

Tabla 19 Descripción del caso de uso Cargar Modelo.

Caso de Uso:	Salvar Modelo
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. Selecciona la opción Salvar Modelo.	1.1. Muestra un cuadro de dialogo solicitando el nombre y la dirección donde desea guardar el modelo.
2. Inserta el nombre y selecciona la dirección donde desea salvar el modelo.	2.1. El sistema guarda en un fichero el modelo.
Pos condiciones	Se salva en un fichero un modelo.

Tabla 20 Descripción del caso de uso Salvar Modelo.

Caso de Uso:	**Ejecutar análisis de observabilidad.**
Actores:	Ingeniero
Propósito	Permitir clasificar variables no medidas en observables, así como mostrar la matriz de ocurrencia y los subconjuntos de asignación detectados por el análisis.
Resumen:	El caso de uso comienza cuando el ingeniero selecciona la opción Ejecutar Análisis de Observabilidad.
Precondiciones	Debe haber sido creada un modelo con anterioridad.
Referencias	R1
Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. El actor selecciona la opción Ejecutar Análisis de Observabilidad.	1.1. El sistema detecta subconjuntos de asignación.

	1.2. Clasifica variables no medidas en observables. 1.3. Muestra la matriz de ocurrencia. 1.4. Muestra los subconjuntos de asignación detectados.
Pos condiciones	Son clasificadas en observables algunas o todas las variables no medidas del modelo.

Tabla 21 Descripción del caso de uso Ejecutar Análisis de Observabilidad.

3. Análisis, Diseño y Desarrollo.

En este capítulo se profundiza más en el desarrollo del sistema, dando continuidad a lo tratado en el capítulo anterior. Diagramas de clases del diseño y de secuencia por cada caso de uso como el resultado del refinamiento de las etapas anteriores.

3.1. Diagrama de clases del diseño.

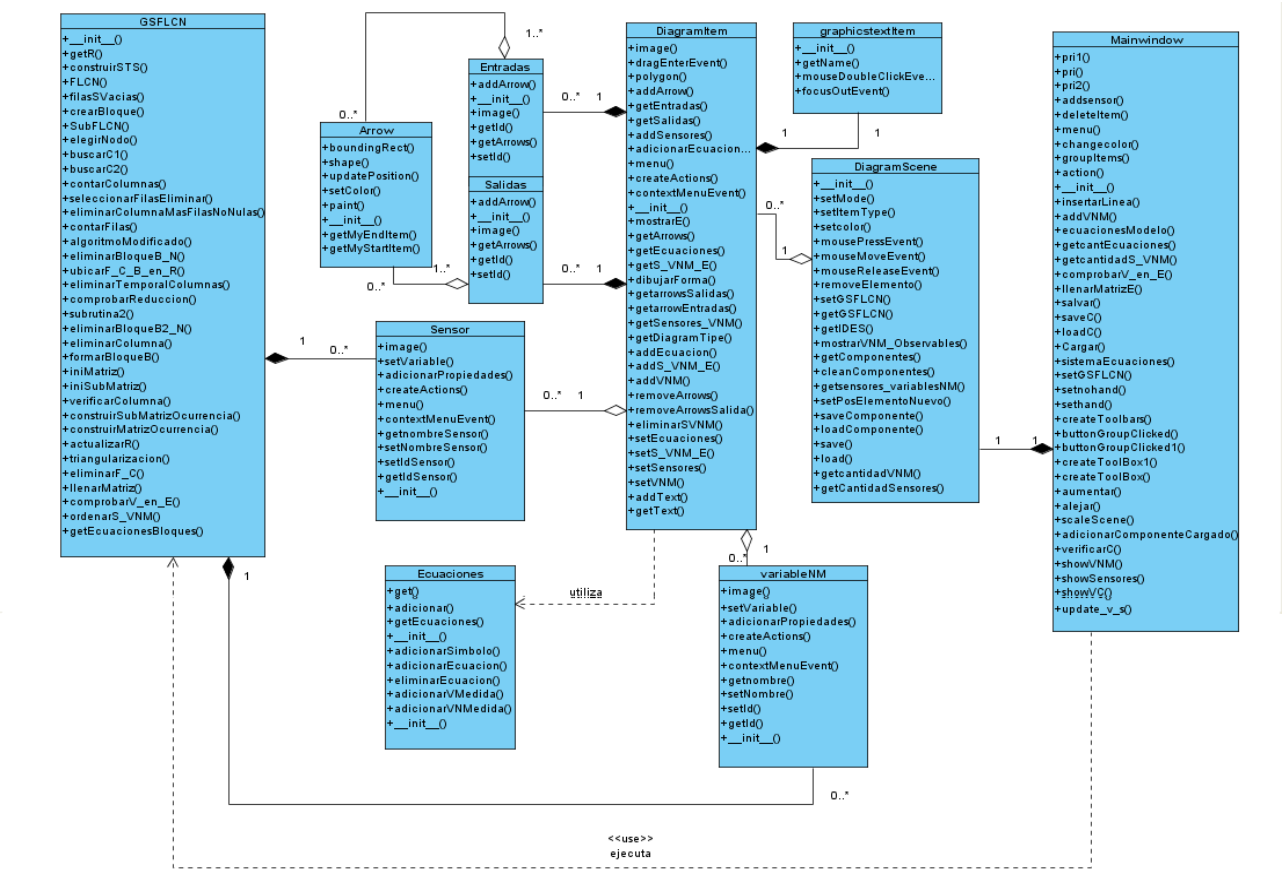


Figura 6 Diagrama de Clases del DSS.

3.1.1. Descripción de las clases del diseño.

Nombre: Mainwindow	
Tipo de Clase: Interfaz	
Operaciones	
Nombre	insertarLinea
Descripción	Función encargada de poner en modo "InsertLine" a la escena.
Nombre	addsensor
Descripción	Función encargada de poner en modo “InsertSensor” a la escena.
Nombre	addVNM
Descripción	Función encargada de poner en modo “InsertVNM” a la escena.
Nombre	deleteItem
Descripción	Función encargada de eliminar elementos de la escena.

Nombre	menu
Descripción	Función encargada de crear el menú.
Nombre	getcantEcuaciones
Descripción	Función encargada de devolver la cantidad de ecuaciones existentes en el modelo.
Nombre	getcantidadS_VNM
Descripción	Función encargada de devolver la cantidad de sensores y variables no medidas del modelo.
Nombre	comprobarV_en_E(idx_e,idx_i)
Descripción	Función encargada de verificar si en la ecuación “x” se encuentra la variable “y”.
Nombre	llenarMatrizE(E)
Descripción	Función encargada de llenar la matriz E
Nombre	salvar
Descripción	Función encargada de crear un QFileDialog mediante el cual se salvará en un fichero una escena.
Nombre	saveC
Descripción	Función encargada de crear un QFileDialog mediante el cual se salvará en un fichero un componente.
Nombre	loadC
Descripción	Función encargada de crear un QFileDialog mediante el cual se cargará de un fichero un componente.
Nombre	Cargar
Descripción	Función encargada de crear un QFileDialog mediante el cual se cargará de un fichero una escena.
Nombre	analisisdeObservabilidad
Descripción	Función encargada de mandar a ejecutar el análisis de observabilidad y luego mostrar los resultados pertinentes.
Nombre	action
Descripción	Función encargada de crear los QAction.
Nombre	setnohand
Descripción	Función encargada de asignarle a view el modo RubberBandDrag.
Nombre	sethand
Descripción	Función encargada de asignarle a view el modo ScrollHandDrag.
Nombre	createToolbars
Descripción	Función encargada de crear la barra de herramientas.
Nombre	buttonGroupClicked
Descripción	Función encargada de poner en modo, y asignar el tipo de componente básico a ser insertado en la escena cuando es seleccionado uno de sus botones.
Nombre	buttonGroupClicked1
Descripción	Función encargada de poner en modo, y asignar el tipo de componente cargado previamente que va a ser insertado en la escena cuando es seleccionado uno de sus botones.
Nombre	createToolBox1
Descripción	Función encargada de crear los botones correspondientes a los componentes que son cargados por el usuario.
Nombre	createToolBox
Descripción	Función encargada de crear los botones correspondientes a los componentes básicos.
Nombre	aumentar
Descripción	Función encargada de aumentar la escena.
Nombre	alejar

Descripción	Función encargada de alejar la escena.
Nombre	adicionarComponenteCargado(componente)
Descripción	Función encargada de adicionar “componente” a la lista de componentes cargados.
Nombre	verificarC(vnm)
Descripción	Función encargada de verificar si la variable “vnm” se encuentra dentro de las variables clasificadas como observables.
Nombre	showVNM
Descripción	Función encargada de mostrar/ocultar las variables no medidas en presentes en la escena.
Nombre	showSensores
Descripción	Función encargada de mostrar/ocultar los sensores presentes en la escena.
Nombre	showVC
Descripción	Función encargada de mostrar/ocultar las variables clasificadas como observables presentes en la escena.
Nombre	update_v_s
Descripción	Función encargada de mantener actualizados los QLabel que muestran la cantidad de variables y sensores de la escena.
Nombre	__init__()
Descripción	Función encargada de inicializar la clase Mainwindow

Tabla 22 Descripción de clases de diseño Mainwindow

Nombre: Arrow	
Tipo de Clase: Entidad	
Operaciones	
Nombre	boundingRect
Descripción	Función encargada de normalizar la posición de Arrow.
Nombre	shape
Descripción	Función encargada de devolver la forma de la flecha cabecera de Arrow.
Nombre	addVNM
Descripción	Función encargada de poner en modo “InsertVNM” a la escena.
Nombre	updatePosition
Descripción	Función encargada de actualizar la posición de Arrow.
Nombre	setColor(color)
Descripción	Función encargada de asignarle color a myColor.
Nombre	paint
Descripción	Función encargada de dibujar la flecha cabecera de Arrow.
Nombre	getMyEndItem
Descripción	Función encargada de devolver el elemento (Entrada) de Arrow.
Nombre	getMyStartItem
Descripción	Función encargada de devolver el elemento (Salida) de Arrow.
Nombre	__init__(startItem,endItem)
Descripción	Función encargada de inicializar Arrow.

Tabla 23 Descripción de clases de diseño Arrow

Nombre: DiagramItem	
Tipo de Clase: Entidad	
Operaciones	
Nombre	image

Descripción	Función encargada de devolver el Qpixmap formado por DiagramItem.
Nombre	addArrow(arrow)
Descripción	Función encargada de adicionarle arrow a la lista de arrows de DiagramItem.
Nombre	adicionarEcuaciones
Descripción	Función encargada de mostrar la forma VentanaEcuaciones.py.
Nombre	getEcuaciones
Descripción	Función encargada de devolver las ecuaciones de DiagramItem.
Nombre	menu
Descripción	Función encargada de crear el menú correspondiente a DiagramItem.
Nombre	createActions
Descripción	Función encargada de crear los QAction.
Nombre	contextMenuEvent
Descripción	Función encargada de mostrar el menú.
Nombre	getS_VNM_E
Descripción	Función encargada de devolver los sensores y variables presentes en las ecuaciones.
Nombre	dibujarForma
Descripción	Función encargada de crear la formar de DiagramItem dependiendo del tipo.
Nombre	getSalidas
Descripción	Función encargada de devolver las salidas.
Nombre	getEntradas
Descripción	Función encargada de devolver las entradas.
Nombre	getarrowsSalidas
Descripción	Función encargada de devolver los Arrows pertenecientes a las salidas.
Nombre	getarrowEntradas
Descripción	Función encargada de devolver los Arrows pertenecientes a las entradas.
Nombre	getSensores_VNM
Descripción	Función encargada de devolver los sensores y variables no medidas.
Nombre	getDiagramTipe
Descripción	Función encargada de devolver el tipo.
Nombre	addEcuacion(ecuacion)
Descripción	Función encargada de adicionar ecuacion a la lista de ecuaciones.
Nombre	addS_VNM_E(Lista)
Descripción	Función encargada de adicionar “Lista” a la lista de sensores y variables no medidas presentes en ecuaciones.
Nombre	addSensores(sensor)
Descripción	Función encargada de adicionar sensor a la lista de sensores.
Nombre	addVNM(VNM)
Descripción	Función encargada de adicionar VNM a la lista de variables no medidas.
Nombre	removeArrows
Descripción	Función encargada de eliminar los Arrows pertenecientes a las salidas de DiagramItem.
Nombre	removeArrowsSalida
Descripción	Función encargada de eliminar los Arrows pertenecientes a las entradas de DiagramItem.
Nombre	eliminarSVNM(s_v)
Descripción	Función encargada de eliminar s_v de la lista de sensores y variables

	no medidas.
Nombre	addText(text)
Descripción	Función encargada de asignarle text a gtext.
Nombre	getText
Descripción	Función encargada de devolver gtext.
Nombre	__init__(diagramTipe)
Descripción	Función encargada de inicializar DiagramItem.

Tabla 24 Descripción de clases de diseño DiagramItem

Nombre: DiagramScene	
Tipo de Clase: Entidad	
Operaciones	
Nombre	__init__()
Descripción	Función encargada de inicializar DiagramScene.
Nombre	getIDES
Descripción	Función encargada de devolver un identificador que será asignado a las entradas y salidas de cada uno de los DiagramItem.
Nombre	mostrarVNM_Observables(listaObservables)
Descripción	Función encargada de mostrar en color azul las variables observables de la escena.
Nombre	setMode(mod)
Descripción	Función encargada de asignarle a DiagramScene el modo en que estará activada.
Nombre	setItemType(type)
Descripción	Función encargada de asignarle a DiagramScene el tipo de DiagramItem que será añadido.
Nombre	getComponentes
Descripción	Función encargada de devolver los componentes presentes en “componentes”.
Nombre	cleanComponentes
Descripción	Función encargada de vaciar “componentes”.
Nombre	getsensores_variablesNM
Descripción	Función encargada de devolver “sensores_variablesNM”.
Nombre	setPosElementoNuevo(posElemento)
Descripción	Función encargada de asignar el tipo de componente cargado que será insertado en la escena.
Nombre	getcantidadVNM
Descripción	Función encargada de devolver la cantidad de variables no medidas presentes.
Nombre	getCantidadSensores
Descripción	Función encargada de devolver la cantidad de sensores.
Nombre	mousePressEvent
Descripción	Función encargada de controlar el evento en que se presione el mouse en la escena.
Nombre	saveComponente(fil)
Descripción	Función encargada de salvar el componente seleccionado en el fichero “fil”.
Nombre	loadComponente(fil,componentes)
Descripción	Función encargada de cargar el componente guardado en el fichero “fil” y ponerlo en la lista de componentes “componentes”.
Nombre	save(fil)
Descripción	Función encargada de guardar una escena en el fichero “fil”.
Nombre	load(fil)

Descripción	Función encargada de cargar una escena almacenada en el fichero “fil”.
Nombre	mouseMoveEvent
Descripción	Función encargada de manipular los eventos en que el mouse es movido por la escena.
Nombre	mouseReleaseEvent
Descripción	Función encargada de manipular los eventos en los que el mouse es liberado después de un clic en la escena.
Nombre	removeElemento(elemento)
Descripción	Función encargada de eliminar componentes.

Tabla 25 Descripción de clases de diseño DiagramScene

Nombre: Entrada	
Tipo de Clase: Entidad	
Operaciones	
Nombre	__init__()
Descripción	Función encargada de inicializar Entrada.
Nombre	image
Descripción	Función encargada de devolver la forma (QPolygonF).
Nombre	addArrow(arrow)
Descripción	Función encargada de adicionar un Arrow.
Nombre	getId
Descripción	Función encargada de devolver el identificador.
Nombre	getArrows
Descripción	Función encargada de devolver los “Arrows”.
Nombre	setId(id)
Descripción	Función encargada de asignar un identificador.

Tabla 26 Descripción de clases de diseño Entrada

Nombre: graphicstextItem	
Tipo de Clase: Entidad	
Operaciones	
Nombre	__init__(name)
Descripción	Función encargada de inicializar graphicstextItem.
Nombre	getName
Descripción	Función encargada de devolver el nombre.
Nombre	mouseDoubleClickEvent
Descripción	Función encargada de manipular el evento “doble clic”.
Nombre	focusOutEvent
Descripción	Función encargada de manipular el evento focusOutEvent.

Tabla 27 Descripción de clases de diseño graphicstextItem

Nombre: GSFLCN	
Tipo de Clase: Entidad	
Operaciones	
Nombre	__init__(E,s_vnm_e,ecuaciones,sensores_variablesNM,profundidad)
Descripción	Función encargada de inicializar GSFLCN.
Nombre	getEcuacionesBloques
Descripción	Función encargada de devolver los bloques de ecuaciones detectados.
Nombre	getR

Descripción	Función encargada de devolver la clasificación obtenida.
Nombre	construirSTS
Descripción	Función encargada de construir la matriz de incidencia.
Nombre	FLCN(n,s,r)
Descripción	Función encargada de llamar a SubFLCN con los valores iniciales.
Nombre	filasSVacias(n,camino,S)
Descripción	Función encargada de verificar si existen filas vacías formadas por un camino.
Nombre	crearBloque(camino,filas)
Descripción	Función encargada de crear un bloque de asignación con un camino y filas dadas.
Nombre	SubFLCN(S,n,nodo,camino,longitud)
Descripción	Función recursiva encargada de buscar subconjuntos de asignación mayores de 4.
Nombre	elegirNodo(nodo)
Descripción	Función encargada de seleccionar el nodo menos conectado y que no ha sido marcado.
Nombre	buscarC1(s2,ecuaciones,vnm)
Descripción	Función encargada de buscar una columna con mayor número de elementos no nulos.
Nombre	buscarC2(listaColumnasAntes,listaColumnasNueva,c1)
Descripción	Función encargada de buscar la columna cuyo número de elementos no nulos fue reducido en 2 en “subrutina2”.
Nombre	contarColumnas(r,ecuaciones,vnm,listaCantE_C)
Descripción	Función encargada de contar la cantidad de valores no nulos presentes en “r” y guardarlos en “listaCantE_C”.
Nombre	seleccionarFilasEliminar(M,columna,ecuacionesModelo)
Descripción	Función encargada de seleccionar las filas que serán eliminadas en la función “eliminarColumnaMasFilasNoNulas”.
Nombre	eliminarColumnaMasFilasNoNulas(M,columna,ecuacionesModelo,vnm,s_VNM_e,svnm)
Descripción	Función encargada de eliminar la columna con más filas no nulas en la función “algoritmoModificado”.
Nombre	contarFilas (r,ecuaciones,listaCantE_F)
Descripción	Función encargada de contar la cantidad de elementos no nulos presentes en “r” y almacenarlos en “listaCantE_F”.
Nombre	algoritmoModificado(n,s3,r)
Descripción	Función encargada de detectar subconjuntos de asignación de tamaño 3.
Nombre	eliminarBloqueB_N(bloqueB3)
Descripción	Función encargada de eliminar de N el conjunto de asignación de tamaño 3 “bloqueB3”.
Nombre	ubicarF_C_B_en_R(bloque3)
Descripción	Función encargada de adicionar en la solución el bloque “bloque3”.
Nombre	comprobarReduccion(vnm,svnm,listaColumnasAntes,listaColumnasNueva,c1)
Descripción	Función encargada de comprobar la reducción de filas después de ejecutar la función “eliminarColumnaMasFilasNoNulas”
Nombre	subrutina2(s2,r)
Descripción	Función encargada de detectar subconjuntos de asignación de tamaño 2.
Nombre	eliminarBloqueB2_N(bloque)
Descripción	Función encargada de eliminar de N el bloque de tamaño 2 “bloque”.

Nombre	eliminarColumna(columna,vnm,svnm)
Descripción	Función encargada de eliminar la columna “columna” de N.
Nombre	formarBloqueB(c1,c2,s2,vnm_copia)
Descripción	Función encargada de formar un bloque de tamaño 2.
Nombre	iniMatriz(e,s_vn)
Descripción	Función encargada de inicializar una matriz con valores 0.
Nombre	iniSubMatriz(listaFilas,listaColumnas)
Descripción	Función encargada de inicializar los valores de una matriz de ocurrencia.
Nombre	verificarColumna(lista,columna)
Descripción	Función encargada de verificar que la una columna ya esta presente en una lista de columnas.
Nombre	construirSubMatrizOcurrencia(n)
Descripción	Función encargada de construir sub matrices de ocurrencia de tamaño n.
Nombre	construirMatrizOcurrencia(m)
Descripción	Función encargada de construir una matriz de ocurrencia y almacenarla en “m”.
Nombre	actualizarR
Descripción	Función encargada de actualizar la solución.
Nombre	triangularizacion(matriz)
Descripción	Función encargada de triangularizar la matriz de ocurrencia “matriz”.
Nombre	eliminarF_C(i,posV,matriz)
Descripción	Función encargada de eliminar una fila y columna de N.
Nombre	llenarMatriz(s,eM,s_v_e,s_vnm)
Descripción	Función encargada de llenar la matriz solución “s”.
Nombre	comprobarV_en_E(i,e,s_v_e,s_vnm)
Descripción	Función encargada de chequear si una variable no medida se encuentra en una ecuación determinada.
Nombre	ordenarS_VNM
Descripción	Función encargada de ordenar las variables no medidas primero que las variables medidas (Sensor).

Tabla 28 Descripción de clases de diseño GSFLCN

Nombre: Salidas	
Tipo de Clase: Entidad	
Operaciones	
Nombre	**__init__()**
Descripción	Función encargada de inicializar Salidas.
Nombre	image
Descripción	Función encargada de devolver la forma (QPolygonF).
Nombre	addArrow(arrow)
Descripción	Función encargada de adicionar un Arrow.
Nombre	getId
Descripción	Función encargada de devolver el identificador.
Nombre	getArrows
Descripción	Función encargada de devolver los “Arrows”.
Nombre	setId(id)
Descripción	Función encargada de asignar un identificador.

Tabla 29 Descripción de clases de diseño Salidas

Nombre: Sensor	
Tipo de Clase: Entidad	
Operaciones	
Nombre	__init__()
Descripción	Función encargada de inicializar Sensor.
Nombre	image
Descripción	Función encargada de devoler el Qpixmap formado por Sensor.
Nombre	adicionarPropiedades
Descripción	Función encargada de mostrar una ventana de propiedades.
Nombre	createActions
Descripción	Función encargada de crear los QAction.
Nombre	menu
Descripción	Función encargada de adicionar al menú los QAction creadas en la función “createActions”
Nombre	contextMenuEvent
Descripción	Función encargada de ejecutar el menú.
Nombre	getnombreSensor
Descripción	Función encargada de devolver el nombre del sensor.
Nombre	setNombreSensor(nombre)
Descripción	Función encargada de asignarle al sensor en nombre “nombre”.
Nombre	setIdSensor(id)
Descripción	Función encargada de asignar al sensor el identificador “id”.
Nombre	getIdSensor
Descripción	Función encargada de devolver el identificador del sensor.

Tabla 30 Descripción de clases de diseño Sensor

Nombre: variableNM	
Tipo de Clase: Entidad	
Operaciones	
Nombre	__init__()
Descripción	Función encargada de inicializar variableNM.
Nombre	image
Descripción	Función encargada de devoler el Qpixmap formado por variableNM.
Nombre	adicionarPropiedades
Descripción	Función encargada de mostrar una ventana de propiedades.
Nombre	createActions
Descripción	Función encargada de crear los QAction.
Nombre	menu
Descripción	Función encargada de adicionar al menú los QAction creadas en la función “createActions”
Nombre	contextMenuEvent
Descripción	Función encargada de ejecutar el menú.
Nombre	getnombre
Descripción	Función encargada de devolver el nombre de la variables no medida.
Nombre	setNombre(nombre)
Descripción	Función encargada de asignarle a la variable no medida el nombre “nombre”.
Nombre	setId (id)
Descripción	Función encargada de asignar a la variable no medida el identificador “id”.
Nombre	getId
Descripción	Función encargada de devolver el identificador de la variable no

	medida.

Tabla 31 Descripción de clases de diseño variableNM

Nombre: Ecuaciones	
Tipo de Clase: Interfaz	
Operaciones	
Nombre	__init__(Title,ecua,sensores,VNM,S_VNM_E,resultadoGSFLCN)
Descripción	Función encargada de inicializar Ecuaciones.
Nombre	adicionarEcuacion
Descripción	Función encargada de adicionar una ecuación.
Nombre	eliminarEcuacion
Descripción	Función encargada de eliminar una ecuación.
Nombre	adicionarSimbolo
Descripción	Función encargada de adicionar un símbolo a una ecuación.
Nombre	adicionarVMedida
Descripción	Función encargada de adicionar una variable medida (Sensor) a una ecuación.
Nombre	adicionarVNMedida
Descripción	Función encargada de adicionar una variable no medida a una ecuación.

Tabla 32 Descripción de clases de diseño Ecuaciones

3.2. **Diagramas de secuencia.**

En todos los casos siguientes en los que se encuentra la acción triggered() se referirá a la acción sobre el menú correspondiente al caso de uso en cuestión.

3.2.1. Diagrama de Secuencias “Insertar Componente”.

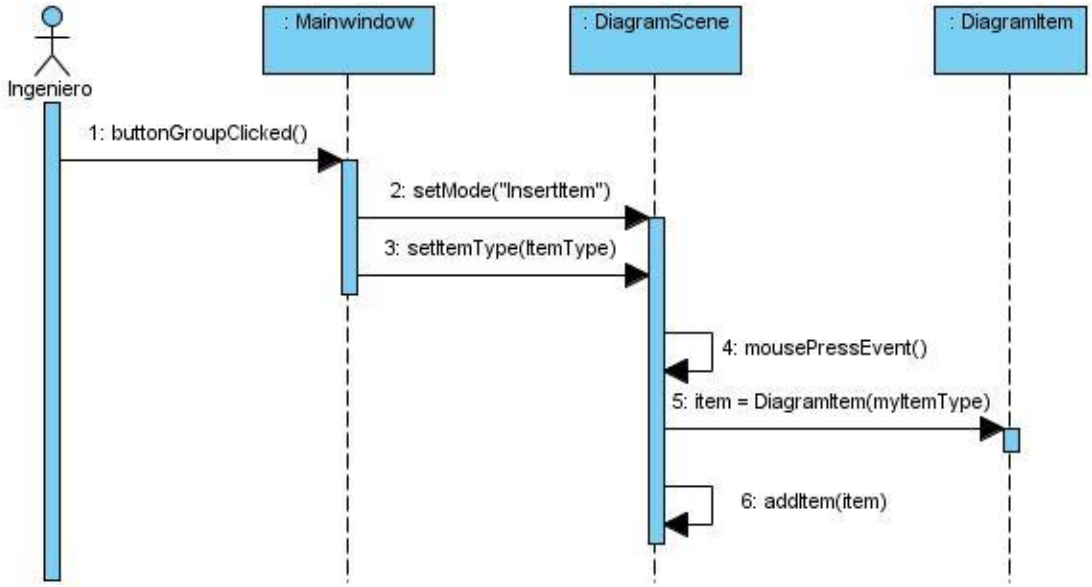


Figura 7 Diagrama de Secuencias "Insertar Componente".

3.2.2. Diagrama de secuencias "Eliminar Componente".

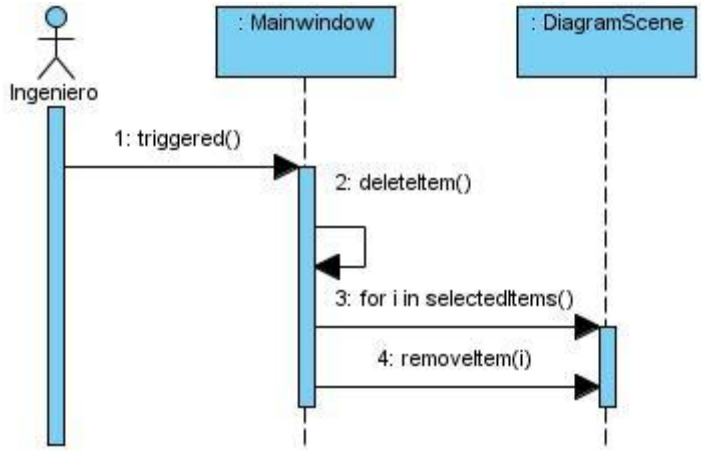


Figura 8 Diagrama de secuencias "Eliminar Componente".

3.2.3. Diagrama de secuencias "Salvar Componente".

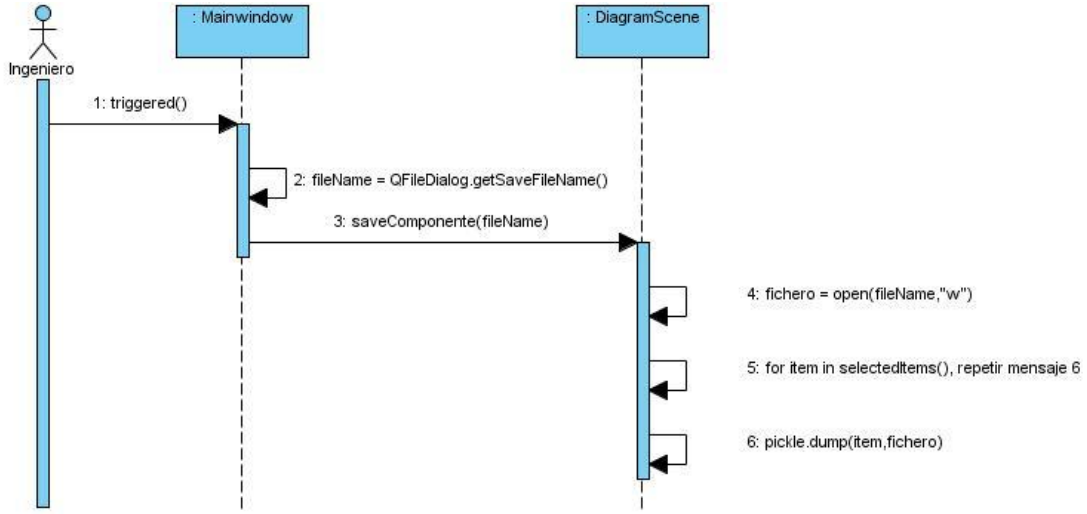


Figura 9 Diagrama de secuencias "Salvar Componente".

3.2.4. Diagrama de secuencias "Cargar componente".

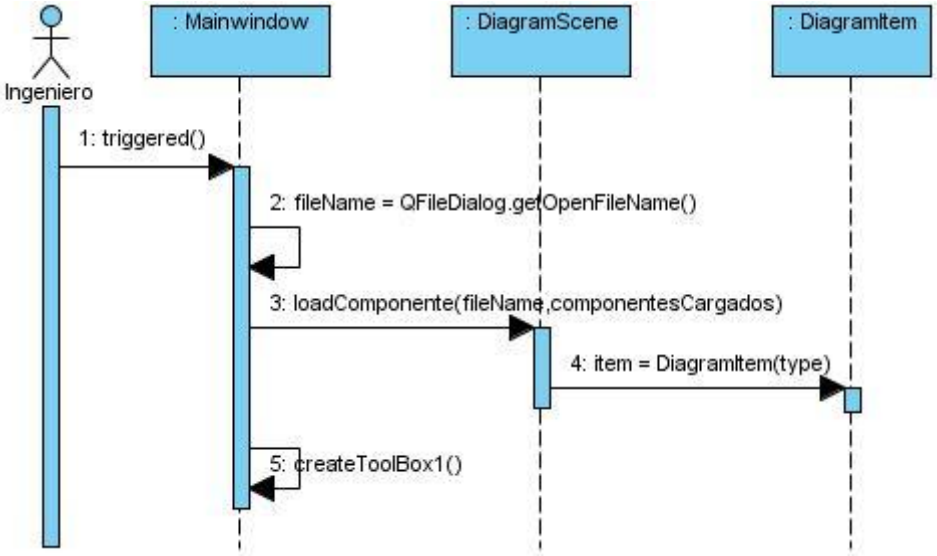


Figura 10 Diagrama de secuencias "Cargar componente".

3.2.5. Diagrama de secuencias "Insertar Sensor".

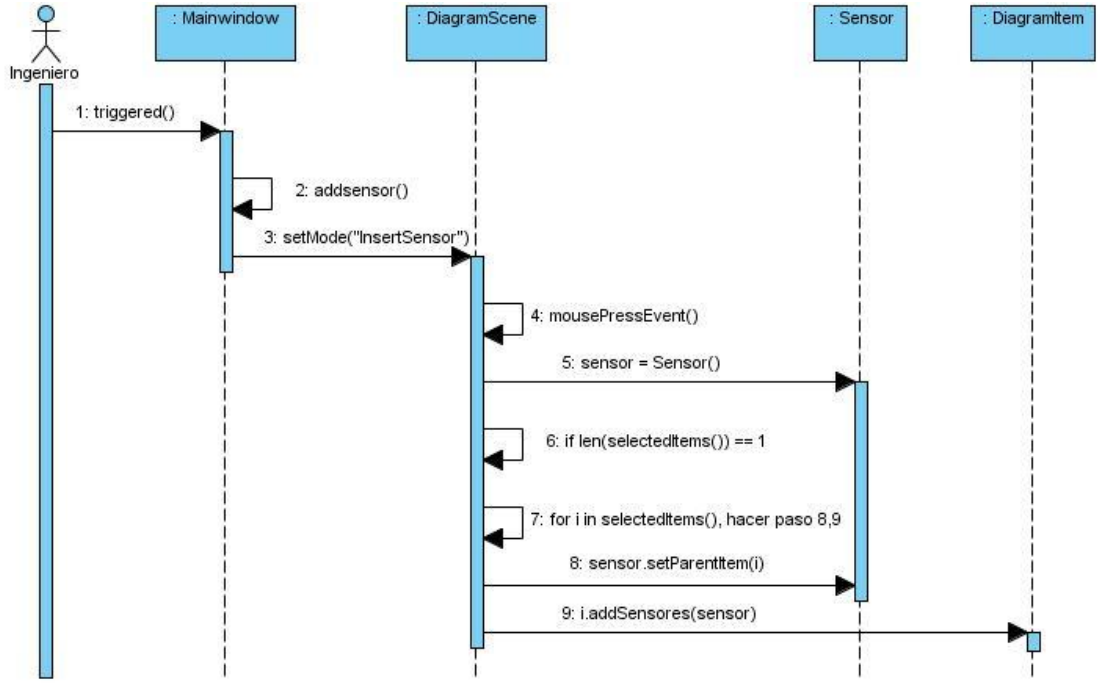


Figura 11 Diagrama de secuencias "Insertar Sensor".

3.2.6. Diagrama de secuencias "Eliminar Sensor".

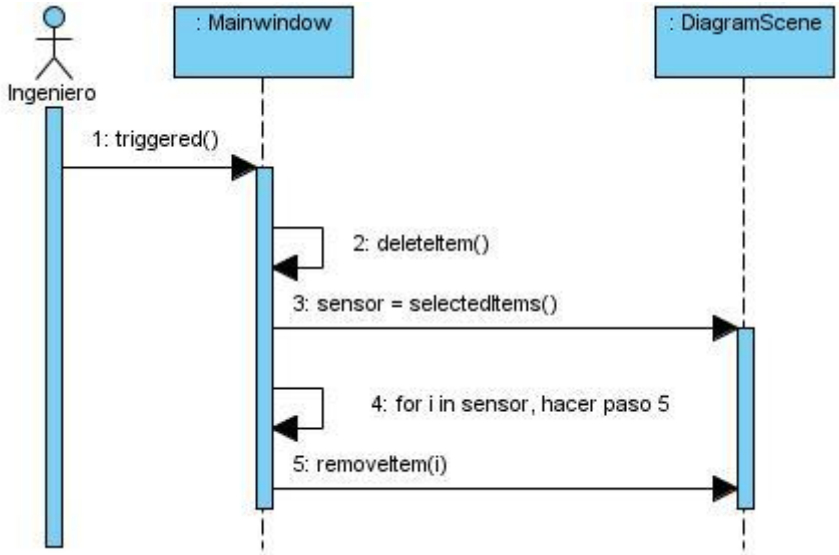


Figura 12 Diagrama de secuencias "Eliminar Sensor".

3.2.7. Diagrama de secuencias "Insertar Variable no Medida".

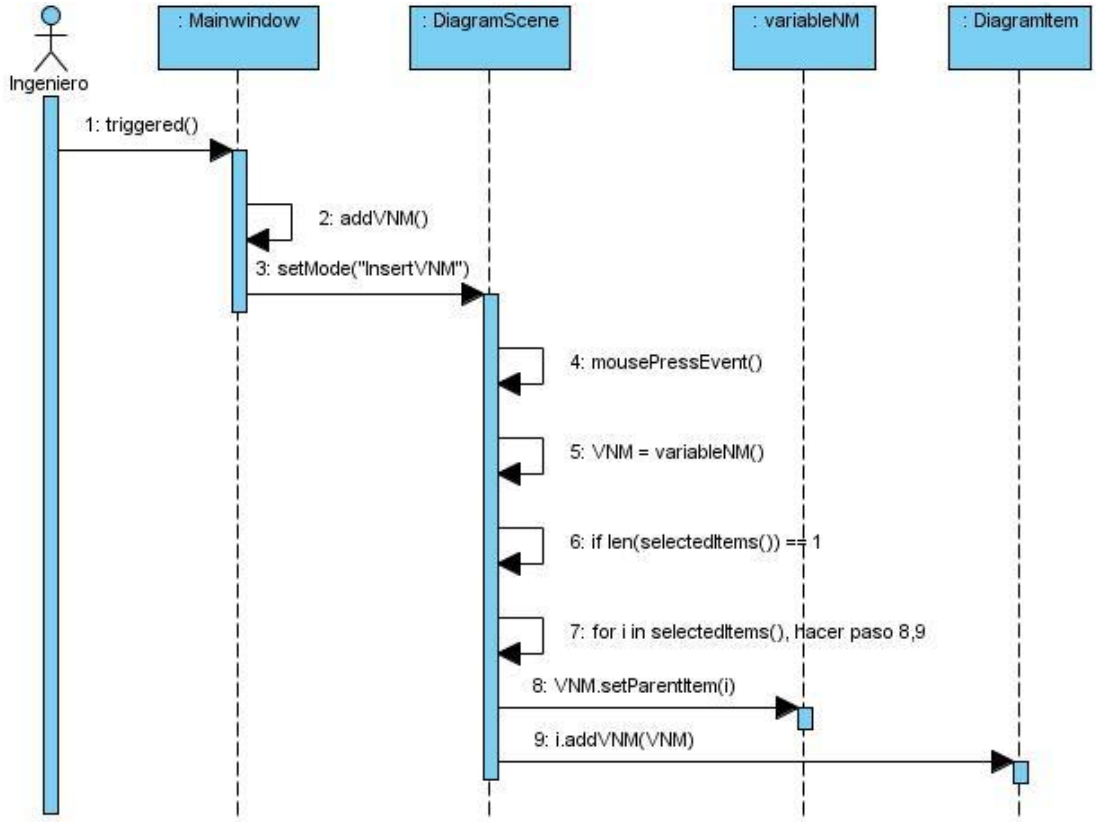


Figura 13 Diagrama de secuencias "Insertar Variable no Medida".

3.2.8. Diagrama de secuencias "Eliminar Variable no Medida".

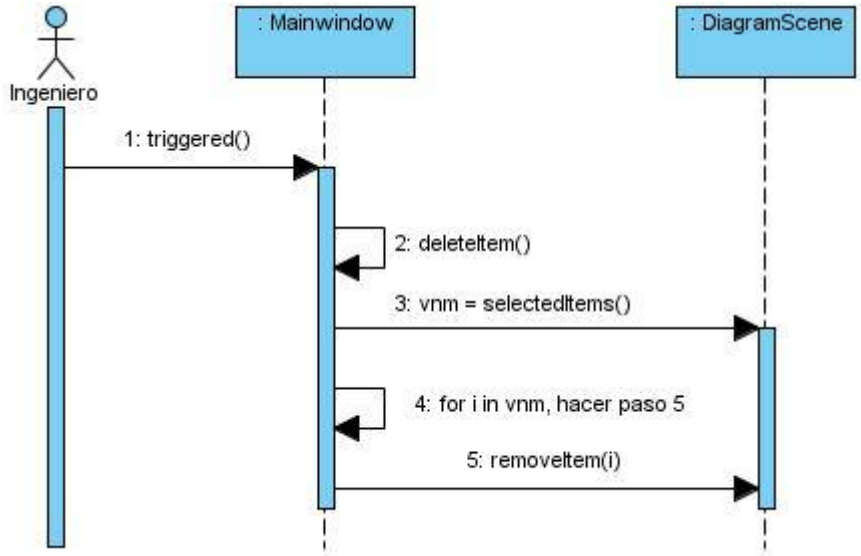


Figura 14 Diagrama de secuencias "Eliminar Variable no Medida".

3.2.9. Diagrama de secuencias "Insertar Corriente".

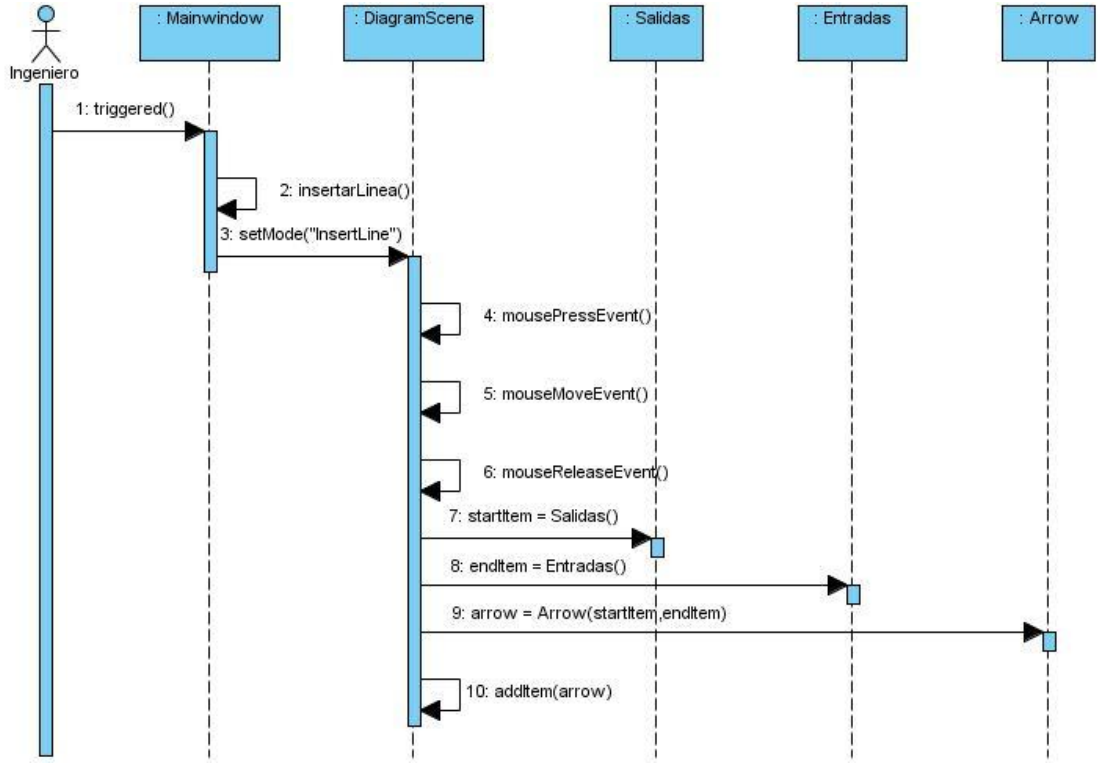


Figura 15 Diagrama de secuencias "Insertar Corriente".

3.2.10. Diagrama de secuencias "Eliminar Corriente".

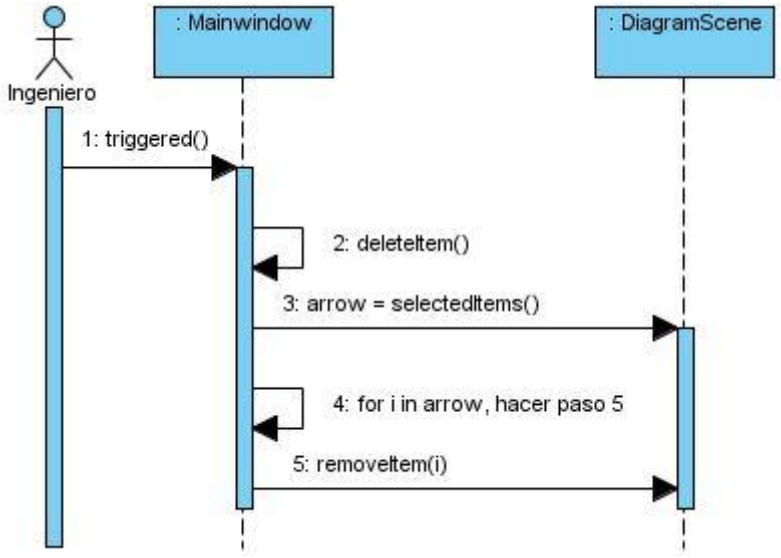


Figura 16 Diagrama de secuencias "Eliminar Corriente".

3.2.11. Diagrama de secuencias "Adicionar Ecuación".

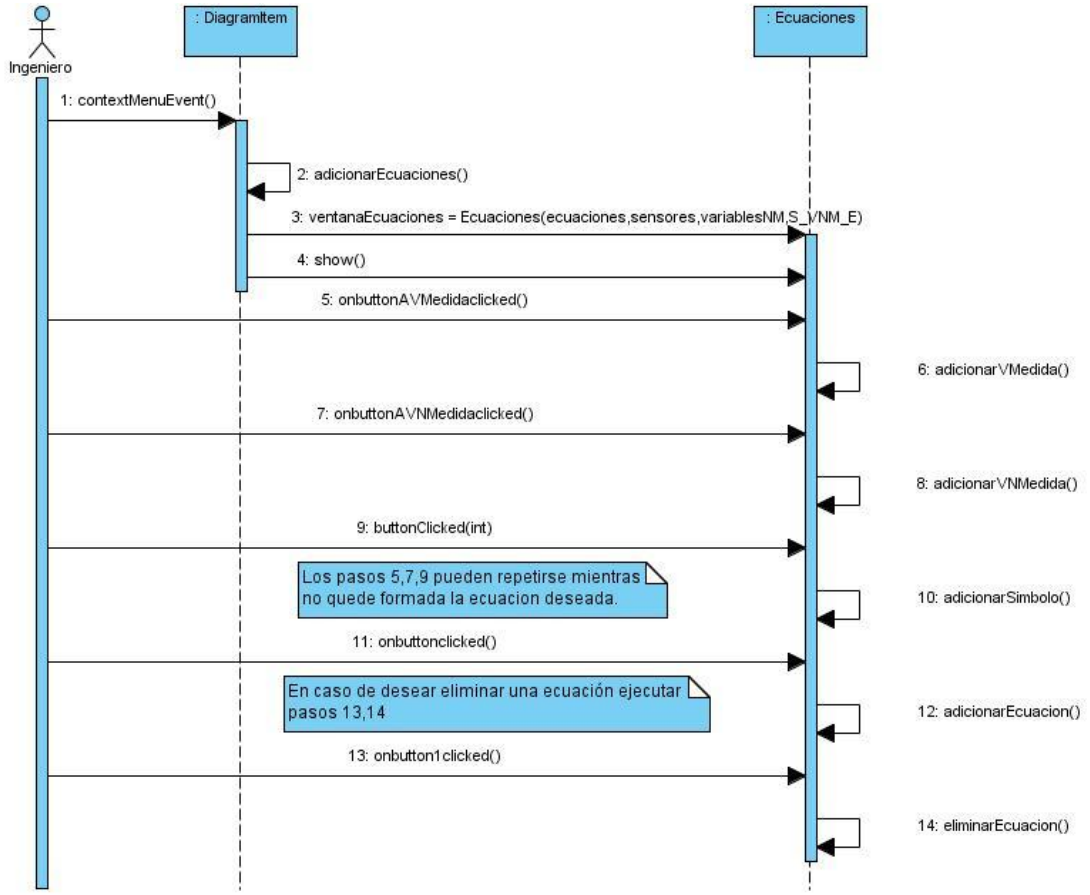


Figura 17 Diagrama de secuencias "Adicionar Ecuación".

3.2.12. Diagrama de secuencias "Ejecutar análisis de observabilidad".

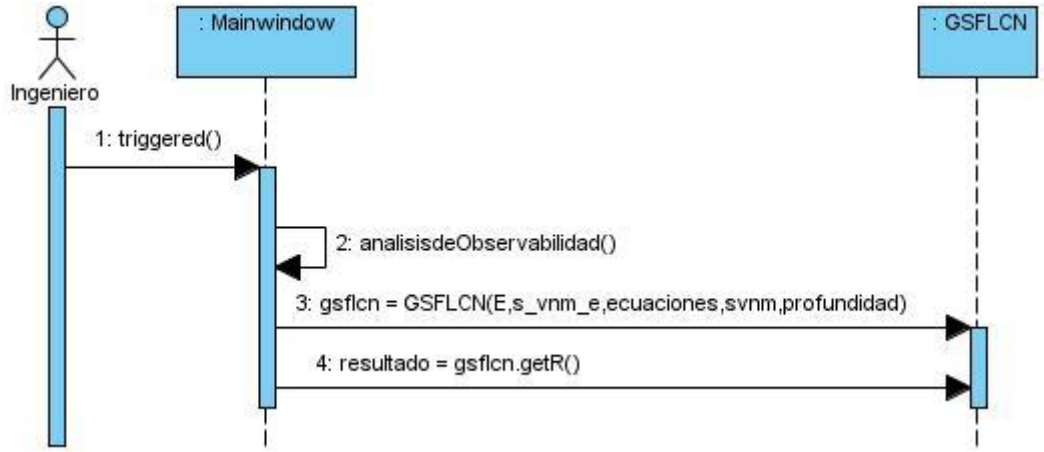


Figura 18 Diagrama de secuencias "Ejecutar análisis de observabilidad".

3.3. **Diagrama de componentes.**

Esta etapa del proyecto constituye el paso del diseño de clases a la creación de componentes físicos, que se traducen en ficheros .py correspondientes a la implementación en PyQt.

3.3.1. Diagrama de componentes del DSS.

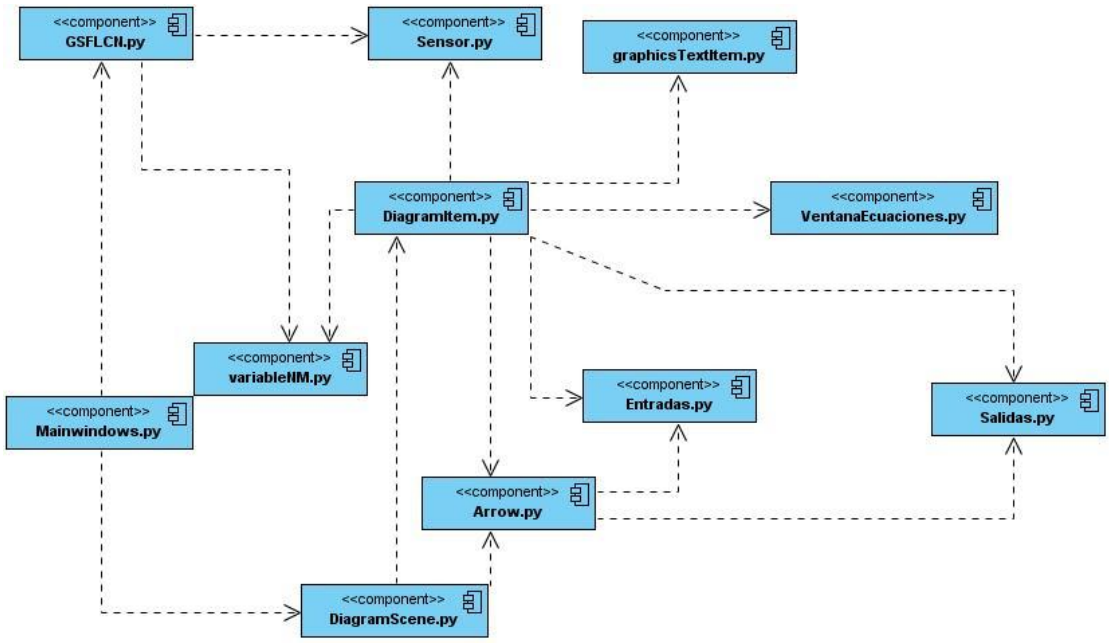


Figura 19 Diagrama de componentes del DSS.

Conclusiones

Con la realización de este trabajo se logró obtener un sistema de soporte de decisión, que permite la creación de modelos de plantas que pueden ser procesados por el algoritmo de observabilidad GS-FLCN, con el cual es posible obtener una clasificación de variables robusta para plantas de pequeño y mediano alcance. La herramienta desarrollada además, ayuda al ingeniero de procesos a realizar un buen diseño de instrumentación dado el alto conocimiento de variables que puede obtener de la misma, tributando esto a la obtención de un conocimiento más confiable de la planta, dándole cumplimiento de esta forma a los objetivos planteados.

Recomendaciones e investigaciones futuras.

Son muchas las recomendaciones que se pueden hacer sobre el sistema desarrollado en este trabajo de diploma dado el amplio rango de investigación a la que puede ser sometido el diseño de instrumentación de plantas industriales, algunas de ellas son las siguientes:

1. Evolución del actual Sistema de Soporte de Decisión hacia un paquete mas completo para diseño de instrumentación, lo que implicaría la incorporación de un módulo para análisis de redundancias y otro para la reconciliación de datos.
2. Agregar otros algoritmos de observabilidad entre los que el ingeniero de procesos pueda escoger dependiendo del alcance del modelo que desee desarrollar.
3. Trabajar en el desarrollo de un módulo que permita la generación automática de ecuaciones.

Glosario de términos.

A

Análisis de observabilidad: Proceso realizado para la detección de variables observables.

B

Balance de masa: Ecuación correspondiente a un componente, esta puede ser de volumen, presión etc.

C

Componentes: Objetos gráficos que se encuentran en la paleta del DSS utilizados para los diseños de los modelos.

Configuración de instrumentación: Proceso que se realiza con el objetivo de obtener un mejor nivel de conocimiento del estado de un proceso.

M

Modelo matemático: Un sistema donde todos los comportamientos u opciones se pueden simular por medio de ecuaciones matemáticas cuyas variables están previamente establecidas de acuerdo a lo que se quiere contemplar.

S

Sensor: Es un dispositivo capaz de transformar magnitudes físicas o químicas, llamadas variables de instrumentación, en magnitudes eléctricas.

V

Variable no medida: Variable del proceso cuyos valores no son sensados regularmente.

Variables no observables: Variables cuyo valor no puede calcularse a partir de las variables medidas mediante ecuaciones del modelo.

Variable observable: Variable del proceso que puede ser calculada a partir de las variables medidas usando ecuaciones del modelo.

Referencias bibliográficas.

[1] Blanchette, J., & Summerfield, M. (2009). *C++ GUI Programming with Qt 3.*

[2] Carolina Olivera, A., Ponzoni, I., & Brignole, N. (2006). re-ingeniería de un paquete de software para el análisis de observabilidad en diseño de instrumentación. *Mecánica Computacional Vol XXV* .

[3] Cusido, A. R. (2000). *Control de Procesos.*

[4] Duque, R. G. (2009). *Python para todos.*

[5] E.Dari, C. R. (1997). An efficient implementation of the first least-connected node strategy . *Mecánica Computacional, Volumen XVIII* .

[6] Flores, O. A. (2007). *Análisis de observabilidad y controlabilidad para sistemas diferencialmente planos.* Universidad Autónoma Metropolitana.

[7] L.F.Acebes, A. R. (2009). Análsis en línea de tiempo del estado energético de plantas azucareras. *Revista Iberoamericana de Automática e Informática Industrial.*

[8] Maldonado, D. (2008). *PyQt, Desarrollando aplicaciones de escritorio.*

[9] Marzal, A., & García, I. (2003). *Introducción a la programacion en Python.*

[10] Miguel A. Lozano, J. A. (2002). Clasificación de variables y reconciliación de datos en ingeniería de procesos. *Revista Internacional de métodos numéricos para cálculo y diseño de ingeniería.*

[11] Pedro, T. d. (2009). Conciliación de datos. *Revista Iberoamericana de Automática e Informática Industrial* .

[12] Penin, A. R. (2006). *Sistemas SCADA.*

[13] Pilgrim, M., Callejo Giménez, F., & Cárdenas Medina, R. (2005). *Inmersión en Python.*

[14] Ponzoni, I. (2001). *Algoritmos para Análisis de Observabilidad.* Bahía Blanca.

[15] Sánchez., J. A. (2006). *Instrumentación y Control Básico de Procesos.*

[16] Vásquez., R. V. (2006). *Estrategias de Análisis y Exploración de Datos Como Soporte a la Operación y Supervisión de Procesos Químicos.* Barcelona.

[17] Vazquez, G. E. (2002). *Procesamiento paralelo distribuido heterogéneo aplicado a la ingeniería de procesos.* Bahía Blanca.

Anexos.

Anexo A: Seudocódigos de algoritmos de observabilidad.

A.1. GSFLCN

```
Datos de entrada: Matriz de ocurrencia de E
Datos de salida: R

1. Construcción de la matriz de ocurrencia N.
2. Triangularización hacia adelante (N,R).
3. n = 2
   3.1 Construcción de la Submatriz de Ocurrencia(n,S).
   3.2 Subrutina 2 (S,R).
       (localización de subconjuntos de asignación de 2x2)
       Si se detecta un subconjunto admisible,
          entonces volver al paso 2.
          si no ir al paso 4.
       fin-si
4. n = 3
   4.1 Construcción de la Submatriz de Ocurrencia (n,S).
   4.2 Algoritmo Modificado (n,S,R)
       (localización de subconjuntos de asignación de 3x3)
       Si se detecta un subconjunto admisible,
         entonces volver al paso 2.
         si no ir al paso 5.
       fin-si
5. n = 4
   5.1 Construcción de la Submatriz de Ocurrencia (n,S).
   5.2 Aplicar el paso 1 de la Subrutina 2.
   5.3 Algoritmo FLCN (n,S,R)
       (localización de subconjuntos de asignación de nxn, n ≥ 4)
       Si se detecta un subconjunto admisible,
          entonces volver al paso 2.
          si no
            Si (n > máximo tamaño de subconjunto),
                 entonces Terminar y retornar la clasificación;
                 si no n = n+1 y volver al paso 5.1.
            fin-si
        fin-si
```

A.2. Subrutina 2.

Datos de entrada: **S**
Datos de salida: **R**

1. **Si S** tiene al menos dos columnas, **entonces**
   Buscar la columna c1 de **S** con mayor número de elementos no nulos. Si hay más de una columna en esta situación, elegir la de menor denominación.
   si no Terminar.
   fin-si.
2. Eliminar c1 de **S**, junto con todas las filas que tienen elementos no nulos en c1.
3. Buscar una columna c2 cuyo número de elementos no nulos se haya reducido en dos luego de aplicar el paso 2.
4. Formar un bloque **B** de orden dos con c1, c2 y las dos filas presentes en ambas columnas.
5. **Si B** aprueba el test de admisión, **entonces**
   Eliminar c2 de **S**. Ubicar las filas y columnas de **B** en las primeras dos filas y columnas libres de la matriz reordenada **R**. Eliminar de la matriz **M** todas las filas y columnas en **B**. Terminar.
   si no Volver al paso 1.
   fin-si.

A.3. Algoritmo Modificado.

Datos de entrada: **S**
Datos de salida: **R**

1. **Si S** tiene al menos n columnas,
   entonces
   Buscar la columna c1 de **S** con mayor número de elementos no nulos. Si hay más de una columna en esta situación, elegir la de menor denominación.
   si no Terminar.
   fin-si.
2. Eliminar c1 de **S**, junto con todas las filas que tienen elementos no nulos en c1.
3. **Si** existen n-1 columnas cuyo número de elementos no nulos ha sido reducido en uno o más,
   entonces

```
        Remover temporalmente de S las n-1 columnas seleccionadas.
        Si n filas quedaron sin elementos,
          entonces
            Remover temporalmente de S dichas n filas.
            Construir un bloque B de orden n con las filas
            y columnas borradas.
            Ir al paso 4.
          si no
            Restaurar las columnas que habían sido removidas
            temporalmente.
            Volver al paso 1.
        fin-si.
      si no Volver al paso 1.
    fin-si.
4.  Si el bloque B es admisible,
      entonces
        Ubicar las filas y columnas de B en las primeras n filas
        y columnas libres de la matriz reordenada R.
        Eliminar de la matriz N todas las filas y columnas en B.
        Terminar.
      si no
        Recobrar las filas y columnas que habían sido borradas
        temporalmente de S.
        Volver al paso 1.
     fin si.
```

A.4. First Least-Connected Node (FLCN).

```
Datos de entrada: S, n, nodo, camino, longitud
Datos de salida: éxito, R



Si longitud = 0,
  entonces
    Si n filas de S quedaron vacías,
      entonces
        Construir un conjunto de asignación B con las n filas
        vacías y las n columnas asociadas a los nodos del camino.
        Aplicar el test de admisión a B.
        Si B es admisible,
          entonces
            Remover de N las filas y columnas de B.
            Agregar las filas y columnas de B a las
            primeras n filas y columnas libres
            de la matriz reordenada R. éxito = verdadero.
          si no éxito = falso.
         fin-si
       si no éxito = falso.
      fin-si
   si no
      finalizar = falso.
      Mientras (no finalizar) y (no éxito) hacer:
        [*]Elegir un nodo sin estampilla adyacente a nodo.
        Si existen varios nodos adyacentes sin marcar,
        seleccionar el menos conectado. Si hay varios nodos con
        el menor nivel de conexión elegir el de menor etiqueta.
        Si no existe un nodo adyacente a nodo sin marcar,
          entonces
            finalizar = verdadero. éxito = falso.
          si no
            nodo-anterior = nodo.
            nodo =  el nodo elegido en [*]. Marcar el nodo.
            FLCN (S,n,nodo,camino,longitud-1,éxito,R)
            Si no éxito,
              entonces
                Remover nodo del camino.
                nodo = nodo-anterior.
            fin-si
         fin-si
      fin-mientras
 fin-si
```

A.5. Método Directo.

D.E.: N (matriz de ocurrencia) y R (restricciones iniciales)
D.S.: N reordenada a FTiB.

Etapa 0. Inicialización.
0.1 Construir el bigrafo G(N)=(R,C,E) asociado a N.

Etapa 1. Descomposición Gruesa.
1.1 Obtener un pareamiento maximal P_m de G(N).
1.2 Clasificar en base a P_m las filas en SR1, SR2, VR, HR y las columnas en SC1, SC2 y HC.
1.3 Para cada *fila especial* que exista, clasificarla como:
VR, si todas sus aristas conducen a columnas de **SC1** ó **SC2**.
SR1, si está conectada a una única columna de **HC** y a ninguna columna de **SC2** (en este caso dicha columna y la fila especial conforman un bloque de 1x1).
SR2, si está conectada a una única columna de **HC** y al menos a una columna de **SC2** (en este caso dicha columna y la fila especial forman un bloque de 1x1).
HR, si está conectada a más de una columna de **HC**.

Etapa 2. Descomposición Fina.
2.1 Asociar el digrafo $G(N_1) = (V,E)$ a la matriz N_1 correspondiente al bloque (SR1,SC1).
2.2 Descomponer $G(N_1)$ en sus componentes fuertes N_{11}, N_{12}, …, N_{1q}. Cada N_{1i} corresponde a un bloque de la diagonal.
2.3 Asociar el digrafo $G(N_2) = (V,E)$ a la matriz N_2 correspondiente al bloque (SR2,SC2).
2.4 Descomponer $G(N_2)$ en sus componentes fuertes N_{21}, N_{22}, …, N_{2p}. Cada N_{2i} corresponde a un bloque de la diagonal.

Etapa 3. Test de Admisión.
3.1 Para cada componente fuerte N_{1i},
3.1.1 Chequear que N_{1i} no esté en el conjunto de restricciones R.
3.1.2 Si N_{1i} está prohibido, ir a la Etapa 4 de Reasignación.
3.1.3 Si la reasignación tuvo éxito, volver a la etapa 2. En caso contrario, ir a la Etapa 5 de Reducción N_{1i}.

3.2 Para cada componente fuerte N_{2i}
3.2.1 Chequear que N_{2i} no esté en el conjunto de restricciones R.
3.2.2 Si N_{2i} está prohibido, ir a la Etapa 5 de Reducción con N_{2i}.
3.3 Ir a la Etapa 6.

Etapa 4. Reasignación de una componente fuerte N_{1i}.

4.1 Buscar una arista (r,c), donde r ∈ VR, c pertenece a las columnas presentes en N_{1i}, (k,c)∈Pm y k no fue reasignada por r anteriormente.

4.2 Si tal arista existe, entonces la reasignación es posible y se deben llevar a cabo las siguientes acciones:

4.2.1 Eliminar de Pm la arista (k,c) donde k es una de las filas de N_{1i}.

4.2.2 Agregar (r,c) a Pm; eliminar k de SR1 y N_{1i}.

4.2.3 Agregar r a SR1 y M_{1i}; eliminar r de VR.

4.2.4 Agregar k a VR.

En caso contrario, la reasignación no es posible.

4.3 Volver al paso 3.1.3.

Etapa 5. Reducción del bigrafo G(M) hasta la componente fuerte N_{ij}.

5.1 Eliminar del bigrafo todas los filas y columnas correspondientes a las componentes fuertes anteriores a N_{ij}- es decir, la N_{kl} con k < i ó k = i y l < j - e incorporarlas a la solución.

5.2 Seleccionar una fila de N_{ij} como *fila especial* (que no haya sido seleccionada anteriormente) y eliminarla de G(N).

5.3 Si todas las filas de N_{ij} fueron seleccionadas anteriormente como *fila especial*, seleccionar dos *filas especiales* entre las filas de N_{ij}, sin importar si haya habían sido elegidas antes.

5.4 Volver a la etapa 1.

Etapa 6. Reordenamiento.

6.1 Reordenar N como sigue:

$[N_{11}, N_{12}, \ldots, N_{1p}, N_{21}, N_{22}, \ldots, N_{2q}, (VR,SC1), (HR,HC)]$

6.2 Fin del Algoritmo.

A.6. CDGH.

Datos de entrada: **H(N)**
Datos de salida: **H(R)**

```
Haux ← H(N); H(R) ← ∅;
ord ← 0;
finaliza ← falso;
Mientras no finaliza hacer:
  Si no EncuentraAutoAristas(Haux,ord,H(R))
    entonces
      heurística ← primera heurística;
      Mientras no EncuentraHGP(Haux,heurística,ord,H(R)) y
                no finaliza hacer:
        Si Ultima(heurística)
           entonces finaliza ← verdadero.
           si no heurística ← Próxima(heurística).
        fin-si
      fin-mientras
  fin-si
fin-mientras
```

A.7. EncuentraAutoAristas.

```
Datos de entrada-salida: Haux, ord, H(R)
Valor retornado por la función:
       verdadero: si encuentra al menos una autoarista,
       falso: en caso contrario.

éxito ← falso.
aristasVisitadas ← ∅.
Mientras queden aristas de Haux sin visitar hacer:
  arista ← una arista de Haux no visitada.
  aristasVisitadas ← aristasVisitadas∪{arista}.
  Si la cardinalidad de arista es 1
    entonces {arista es una autoarista}
      nodo ← Nodo(arista,Haux).
      Si Chequeo1x1(nodo,arista,Haux)
        entonces
          ord ← ord+1;
          AgregarHG1x1(nodo,arista,ord,H(R)).
          RemoverArista(arista,Haux).
          RemoverNodo(nodo, Haux).
          éxito ← verdadero.
      fin-si
  fin-si
fin-mientras
```

A.8. Función Buscar HGP.

```
Datos de entrada: heurística, Restricciones
Datos de entrada-salida: Haux, ord, H(R)
Valor retornado por la función:
       verdadero: si encuentra un hipergrafo parcial,
       falso: en caso contrario.

éxito ← falso.
ok ← verdadero.
pila_nodos ← pila_vacía.
pila_aristas ← pila_vacía.
InicializarEnFalso(estampilla).
InicializarEnFalso(visitado).
InicializarEnFalso(aristas_exp).
Mientras ok hacer:
  ok ← BuscarNodo(Haux,heurística,visitado,nodo);
  Si ok entonces {se ha encontrado un nodo no visitado}
      Poner el nodo en la pila_nodos. éxito ← falso.
      estampilla[nodo] ← verdadero. visitado[nodo] ← verdadero.
      Mientras (no hubo éxito)y(no este vacía pila_nodos) hacer:
        existe ← BuscarNodoAdyacente(nodo,Haux,heurística,
                  estampilla,aristas_exp,nodo_ady,arista).
        Si existe entonces {Se encontró un nodo adyacente}
            Poner la arista en la pila_aristas.
            aristas_exp[arista] ← verdadero.
            Si estampilla[nodo_ady]
              entonces {nodo_ady estaba en la pila,
                        luego existe un ciclo}
                Si TestCiclo(Haux,nodo_ady,pila_nodos,
                             pila_aristas,Restricciones,PH,n)
                  entonces {Se halló un hipergrafo parcial}
                    ok ← falso. éxito ← verdadero.
                    AgregarEnHGNxN(PH,n,ord,H(R));
                  si no {este subconjunto es inadmisible}
                    aristas_exp[Tope(pila_aristas)] ← falso.
                    Desapila pila_aristas.
                fin-si
              si no {nodo_ady no está sobre la pila}
                Poner nodo_ady en la pila_nodos.
                nodo ← nodo_ady. estampilla[nodo] ← verdadero.
                visitado[nodo] ← verdadero.
            fin-si
          si no {No se encontró un nodo adyacente}

            aristas_exp[Tope(pila_aristas)] ← falso.
            Desapila pila_aristas.
            Si no está vacía pila_nodos entonces
                aristas_exp[Tope(pila_aristas)] ← falso.
                Desapila pila_aristas. nodo ← Tope(pila_nodos).
            fin-si
        fin-si
      fin-mientras
  fin-si
fin-mientras
Retornar el valor de éxito.
```

A.9. Función TestCiclo.

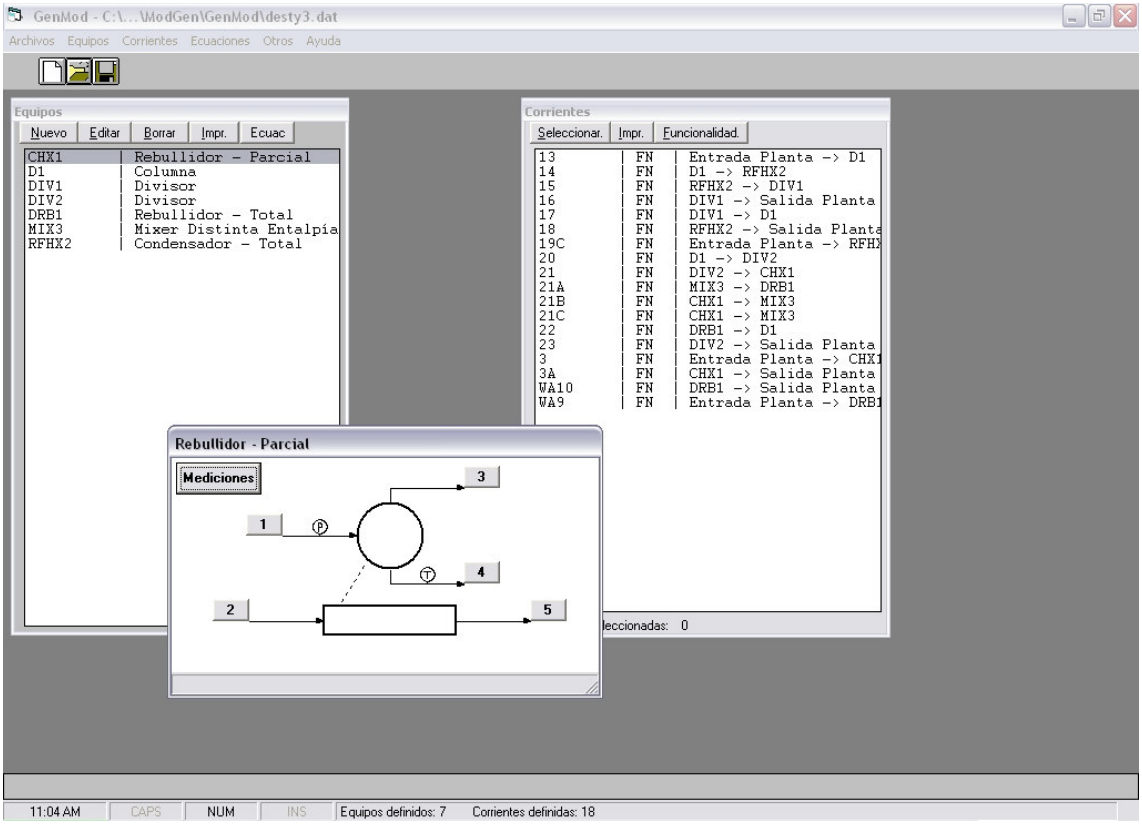
```
Datos de entrada: nodo_ady,pila_nodos,pila_aristas,Restricciones
Datos de entrada-salida: Haux,PH,n
Valor retornado por la función:
       verdadero: si encuentra un subconjunto admisible,
       falso: en caso contrario.

InicializarEnFalso(nodos).
RecuperarPH(nodo_adj,pila_nodos,pila_aristas,PH).
edge ← 1.
éxito ← falso.
aristasVisitadas ← ∅.
Mientras queden aristas de Haux sin visitar hacer:
  arista ← una arista de Haux no visitada.
  aristasVisitadas ← aristasVisitadas∪{arista}.
  Si no Completo(arista, Haux,HP)
    entonces RemoverArista(arista,HP).
  fin-si
fin-mientras
nodosVisitados ← ∅.
Mientras queden nodos de N_Haux sin visitar hacer:
  nodo ← un nodo de N_Haux no visitado.
  nodosVisitados ← nodosVisitados∪{nodo}.
  Si el nodo no pertenece a ninguna arista de HP
    entonces RemoverNodo(nodo,PH).
  fin-si
fin-mientras
Si Orden(HP) = a la cardinalidad de N_HP
  entonces { HP es un hipergrafo parcial cuadrado de Haux}
    Si ChequearNxN(PH,n,Restricciones)
      entonces
        éxito ← verdadero.
        n ← Orden(HP).
    fin-si
fin-si
Retornar el valor de éxito.
```

Anexo B: DSSs.

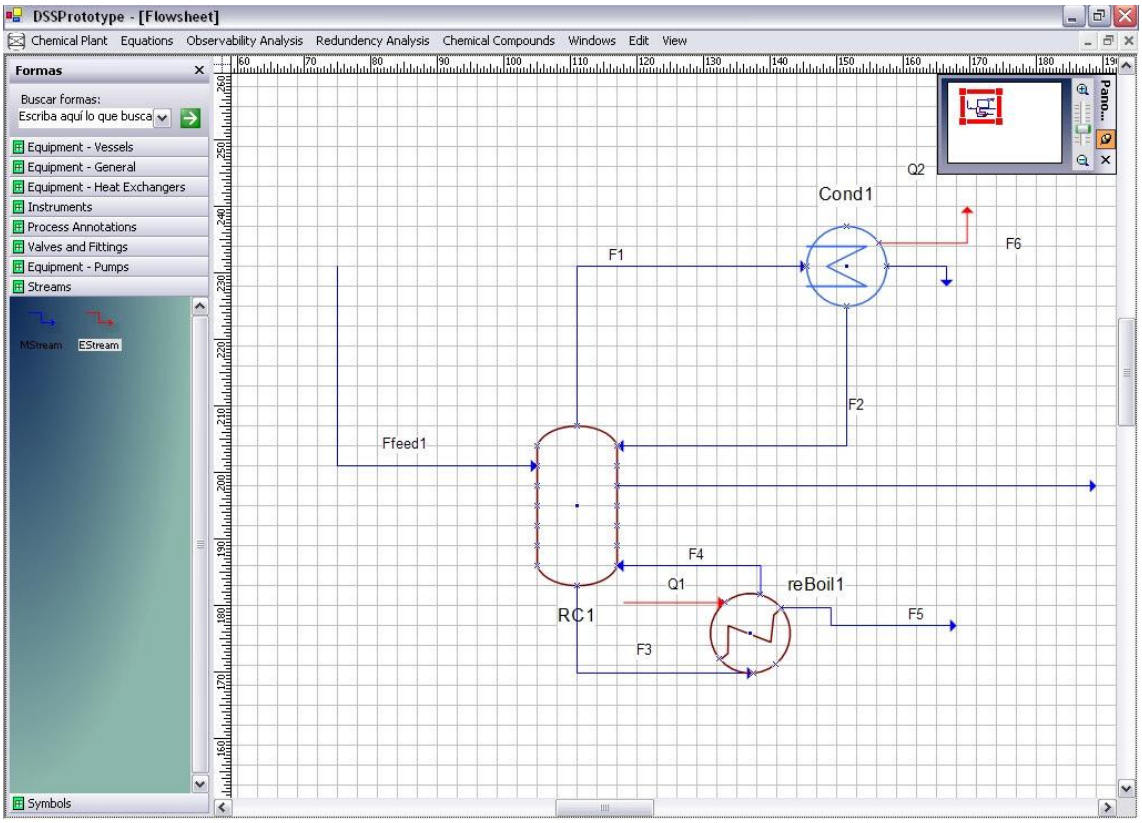
B.1. Interfaz gráfica, definición de equipos y corrientes.

Equipos y corrientes ingresados por el usuario correspondiente a la primera interfaz tratada en el epígrafe 1.4.

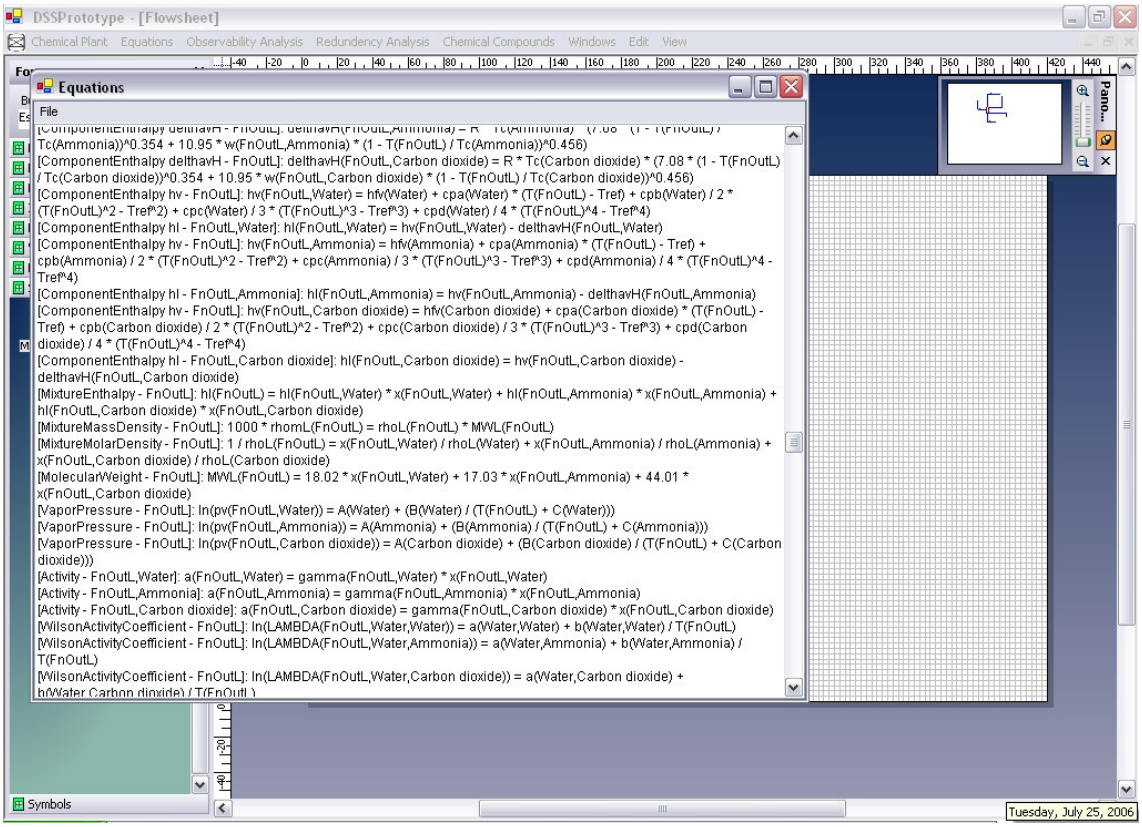


B.2. Interfaz gráfica, pantalla principal.

Pantalla principal del DSS logrado después de la reingeniería realizada al sistema mostrado en el anexo anterior (Anexo B1).



B.3. Pantalla con ecuaciones del modelo.

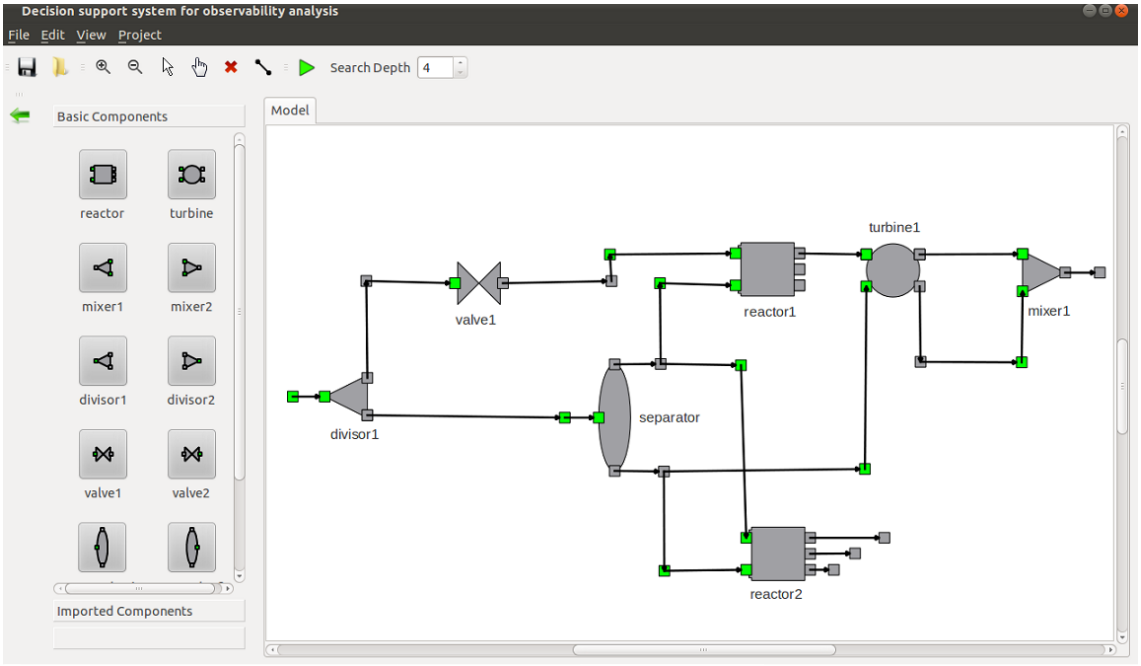


Anexo C: Imágenes de la aplicación.

A continuación se muestran algunas imágenes de la aplicación.

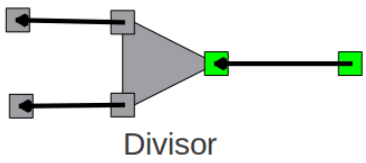
C.1. Pantalla principal + Modelo Hipotético.

Pantalla principal del sistema de soporte desarrollado en el cual se muestra un modelo de una planta no real donde se usan diferentes equipos considerados para la aplicación desarrollada en esta tesis.



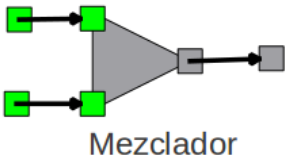
C.2. Divisor.

Es un componente que puede ser usado para dividir una corriente.



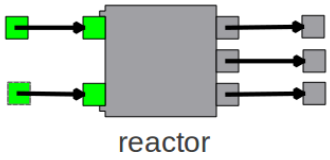
C.3. Mezclador.

Es un componente que puede ser usado para mezclar corrientes de entrada.



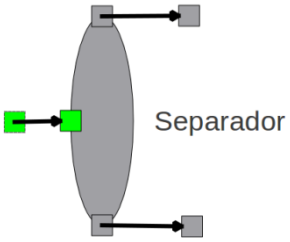
C.4. Reactor.

Es un componente en el cual se producen reacciones químicas para obtener productos de interés.



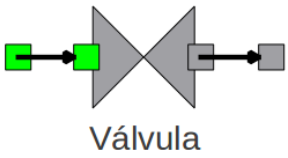
C.5. Separador.

Es un componente en el cual se separa la alimentación en dos fases.



C.6. Válvula.

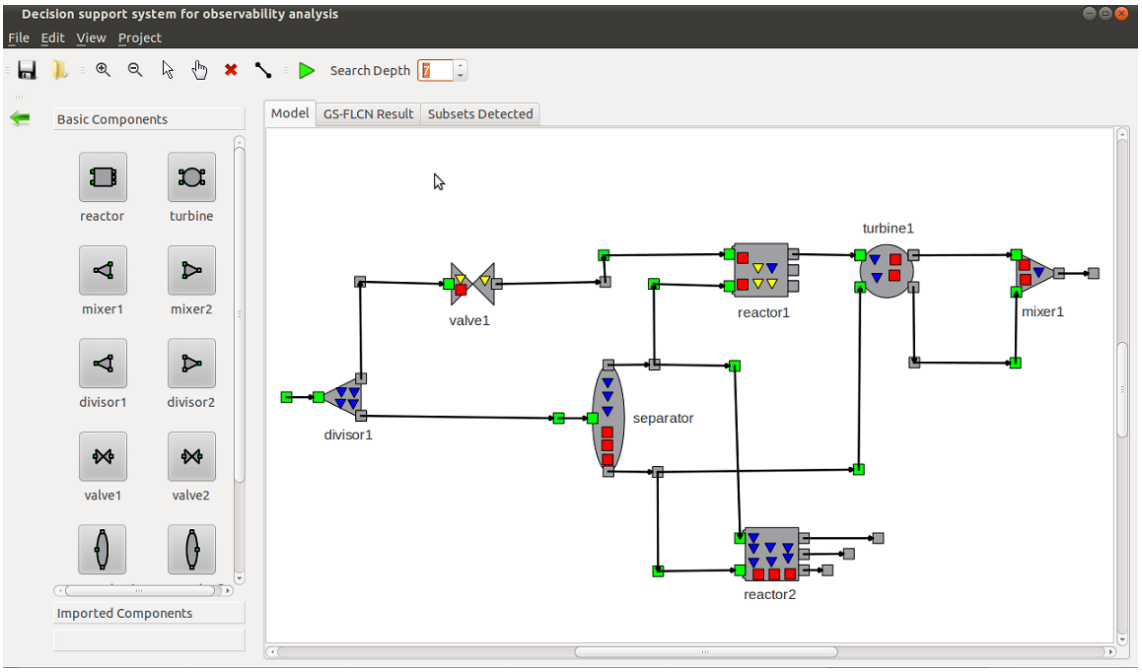
Es un componente para ofrecer resistencia al flujo.



C.7. Pantalla principal + Modelo + Sensores + Variables no medidas.

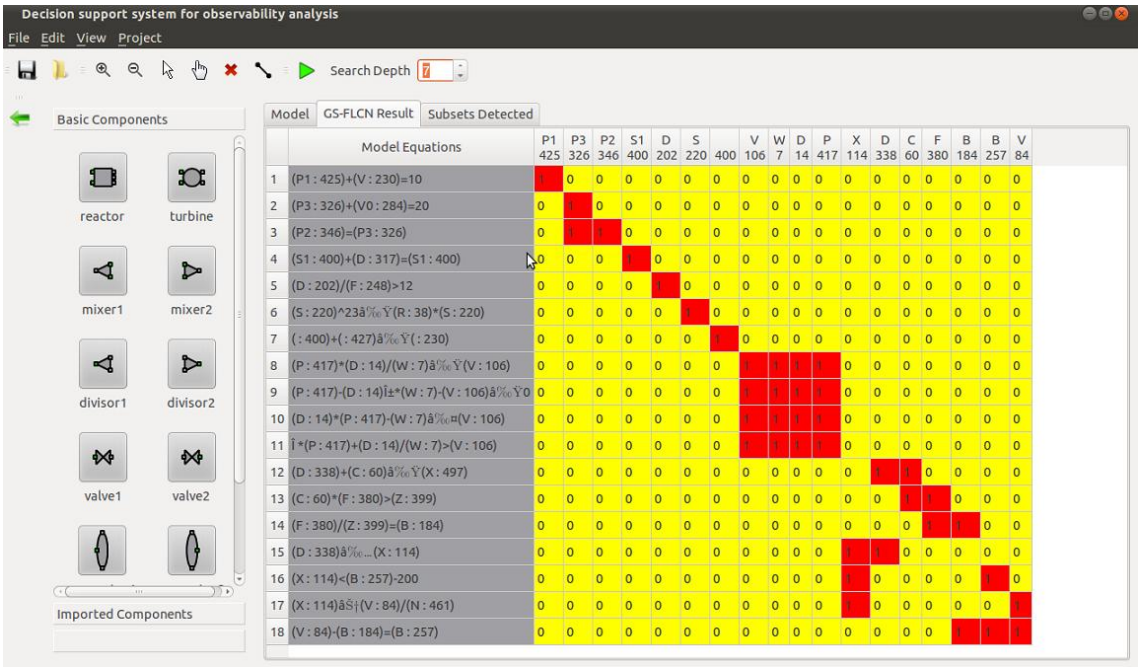
Pantalla principal del DSS más un modelo no real al cual se le añadió un conjunto de sensores , variables no medidas , y ecuaciones.

También se puede observar que después de ejecutado el algoritmo de clasificación de variables las que fueron marcadas como observables son mostradas en diferente color .



C.8. Matriz inferior triangular en bloque.

Matriz obtenida después de ejecutar la clasificación de variables con una profundidad igual 7, en la cual se pueden observar los subconjuntos detectados.



C.9. Subconjuntos de asignación detectados.

Pantalla donde se muestra el número de subconjunto detectado seguido del tamaño del bloque, y las ecuaciones presentes en el mismo.