Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA (SEGOVIA)

Grado en Ingeniería Informática de Servicios y Aplicaciones

Entorno integrado para la simulación de modelos con aplicación a la diagnosis

---

Alumna: Sandra Sastre Muñoz

Tutores: Aníbal Bregón Bregón
José Belarmino Pulido Junquera

Me gustaría expresar mis más sincero agradecimiento

a mis tutores por haberme ayudado
durante este trabajo,

a mi familia y a mi novio por haberme
apoyado durante estos años,

a mis amigos con los que he compartido
mis mejores y peores momentos.

Resumen

Una de las principales líneas de investigación del Grupo de Sistemas Inteligentes del Departamento de Informática es la relacionada con la diagnosis y prognosis de fallos en sistemas industriales y aeroespaciales. Actualmente, se dispone de una serie de modelos de simulación y una serie de algoritmos para la obtención de resultados de diagnosis. Estos algoritmos, implementados utilizando Matlab, no están relacionados entre sí, y es necesario la ejecución del entorno de simulación Matlab, y su profundo conocimiento para ejecutar manualmente cada uno de los modelos de simulación o algoritmos de diagnosis. En este TFG se ha llevado a cabo la creación de una aplicación en Java que a través de la interfaz permite al usuario seleccionar y configurar todos los modelos/algoritmos/parámetros de los experimentos de simulación y diagnosis. La aplicación es capaz de conectarse internamente con Matlab, utilizándolo como "motor de simulación", para obtener los resultados correspondientes a los experimentos de simulación y diagnosis. Gestiona todo el intercambio de información con Matlab. Además, la aplicación muestra y permite al usuario guardar los resultados obtenidos.

Índice de contenido

Índice de tablas

Índice de ilustraciones

Capítulo 1
Introducción

Los sistemas industriales y aeroespaciales son sistemas digitales cada vez más sofisticados, lo cual lleva implícita la utilización de tecnología cada vez más compleja. Este crecimiento en la complejidad de los sistemas provoca que la probabilidad de que se presente un fallo sea cada vez mayor.

La aparición de fallos en estos sistemas puede llegar a tener consecuencias catastróficas. Por ello, deben ser capaces de detectar e identificar estos fallos para evitar lo antes posible sus consecuencias. La simulación y la diagnosis son muy eficaces para la detección e identificación de fallos.

La diagnosis de fallos en sistemas industriales y aeroespaciales es una de las principales líneas de investigación del Grupo de Sistemas Inteligentes del Departamento de Informática. En la actualidad, se dispone de una serie de modelos de simulación y una serie de algoritmos para la obtención de resultados de diagnosis. Estos algoritmos son implementados mediante la utilización de Matlab. Además, al no encontrarse

relacionados entre sí, es necesaria la ejecución del entorno de simulación Matlab y un profundo conocimiento de éste para poder ejecutar manualmente cada uno de los modelos de simulación o algoritmos de diagnosis.

En este TFG se implementará un entorno integrado para la simulación de modelos y la realización de experimentos de diagnosis. Dicho entorno integrado contará con una interfaz amigable y los mecanismos necesarios para controlar dentro del entorno todos los procesos de simulación, realizar experimentos de diagnosis y facilitar la visualización y exportación de resultados.

1.1. Motivación

En los últimos años la tecnología ha evolucionado mucho y cada vez tiene más importancia en las grandes empresas y en el día a día de las personas, como teléfonos móviles, ordenadores, televisores, lavadoras, etc.

Como se ha comentado antes, la probabilidad de que aparezca un fallo en los aparatos tecnológicos es cada vez mayor. Para poder evitar cuanto antes las posibles consecuencias de esos fallos se ha decidido realizar este TFG. Esto será posible, facilitando tanto la ejecución de experimentos con los algoritmos existentes, como la visualización directa de los resultados de simulación y diagnosis.

1.2. Objetivos

El objetivo principal de este TFG es desarrollar con Java un entorno integrado para la simulación de modelos con aplicación a la diagnosis. Esto se lleva a cabo mediante ciertos sub-objetivos:

- El primer sub-objetivo es que la aplicación permita al usuario seleccionar y configurar todos los modelos, algoritmos y parámetros del experimento de simulación, así como ejecutar dicho experimento de simulación.

- El segundo sub-objetivo es que la aplicación permita al usuario seleccionar y configurar todos los modelos, algoritmos y parámetros del experimento de diagnosis y así poder ejecutar dicho experimento de diagnosis.

- El tercer sub-objetivo es que la aplicación debe ser capaz de conectarse internamente con Matlab para utilizarlo como un "motor de simulación". Además, debe ser responsable de gestionar todo el intercambio de información, de manera que pueda capturar instantáneamente los resultados de la ejecución de los algoritmos en Matlab.

- El cuarto sub-objetivo es que la aplicación muestre al usuario los resultados de los experimentos de simulación y diagnosis. Estos resultados se mostrarán en forma de tabla y de gráfica.

- El quinto sub-objetivo es que la aplicación permita al usuario ejecutar el cálculo de residuos, después de seleccionar y configurar los algoritmos y parámetros.

- El sexto y último sub-objetivo es que la aplicación permita al usuario seleccionar y configurar los algoritmos y parámetros para calcular el aislamiento de fallos.

1.3. Organización del documento

En esta sección se va a describir la estructura de este documento. El documento se divide en nueve capítulos y dos apéndices:

- **Capítulo 1: Introducción.** Este capítulo es en el que nos encontramos. En él se presenta el problema y objetivos del TFG, así como la estructura del documento.

- **Capítulo 2: Estado del arte.** En este capítulo se pretende facilitar al usuario la comprensión del documento mediante una descripción del entorno teórico del TFG.

- **Capítulo 3: Plan de proyecto.** En este capítulo se muestra la metodología y el software utilizados, se describe brevemente Java SE y se muestran las estimaciones, la planificación temporal y el presupuesto.

- **Capítulo 4: Análisis.** Este capítulo engloba la identificación de los actores, los requisitos y los diagramas de casos de uso, de clases de análisis y de secuencia.

- **Capítulo 5: Diseño.** En este capítulo se muestra el diagrama de clases de diseño, la arquitectura software y el diseño de la interfaz.

- **Capítulo 6: Pruebas.** En este capítulo se engloban las pruebas realizadas durante el desarrollo de la aplicación.

- **Capítulo 7: Manuales.** Este capítulo contiene los manuales de instalación y de usuario.

- **Capítulo 8: Conclusiones.** Este capítulo trata de plasmar las conclusiones obtenidas tras la realización del TFG y el trabajo futuro que se podría desarrollar para futuras ampliaciones.

- **Capítulo 9: Bibliografía.** En este capítulo se encuentra la relación de libros y escritos consultados para la realización del TFG.

- **Apéndice A: Contenido del CD-ROM.** En este apéndice se presenta la estructura del contenido del CD-ROM que se entrega junto a este documento.

- **Apéndice B: Glosario.** Este apéndice engloba las definiciones de las palabras técnicas que aparecen en este documento.

Capítulo 2
Estado del arte

En este capítulo se realiza una breve descripción del contexto teórico en el que se encuadra este TFG. Mediante esta información se pretende facilitar al usuario unos conocimientos elementales para comprender el contenido de este documento.

2.1.Fallos y diagnosis

A lo largo del tiempo se han ido creando sistemas cada vez más complejos, por lo que esto conlleva un aumento en las formas que puede fallar un sistema. La aparición de fallos en estos sistemas puede tener consecuencias graves (daños materiales o humanos, pérdida de calidad de un producto, ...). Estos fallos pueden deberse al atasque o desgaste de alguna pieza, a la fuga de alguna tubería, etc. Para poder evitar las posibles consecuencia lo antes posible, es necesario detectar que ha ocurrido un fallo e identificarlo.

Entendemos por un **fallo** como la desviación del comportamiento real de un sistema con su comportamiento esperado. Existen diferentes criterios para clasificar los fallos:

- Considerando el **modelo de proceso**:
  - **Fallos aditivos:** entradas desconocidas que actúan en el sistema, presentando un valor nulo normalmente. Cuando se muestran provocan cambios en las salidas del sistema, indistintamente de las entradas conocidas.
  - **Fallos multiplicativos:** cambios en algunos parámetros del sistema. Cuando aparecen provocan cambios en la salida del sistema, los cuales dependen de las entradas conocidas.
- Considerando su **dependencia temporal**:
  - **Fallos abruptos:** sus efectos aparecen de forma inesperada.
  - **Fallos incipientes:** sus efectos aparecen lenta y progresivamente.
  - **Fallos intermitentes:** sus efectos aparecen y desaparecen a lo largo del tiempo.
- Considerando el **componente al que afecta**:
  - **Fallo en sensor:** las lecturas del sensor contienen errores. Las propiedades de la planta no se ven dañadas.
  - **Fallo en actuador:** interrupción o modificación de la influencia del controlador en el sistema. Las propiedades de la planta no se ven dañadas.
  - **Fallo en el proceso:** cambia las propiedades dinámicas de entrada/salida del sistema.

El objetivo principal es conseguir que los sistemas sean tolerantes a fallos, es decir, que sean capaces de reducir sus consecuencias en la medida de lo posible. Para ello, necesitamos realizar el **diagnóstico de fallos**, por lo tanto, debemos detectar y localizar la ocurrencia de un fallo e intentar reducir sus consecuencias. La diagnosis determina por qué un sistema diseñado correctamente no está funcionando como se esperaba. La diagnosis es un proceso iterativo que se divide en tres etapas:

- **Detección de fallos:** determina si ha ocurrido un fallo o no en el sistema.
- **Aislamiento o localización de fallos:** encuentra los componentes donde se localiza el fallo. Consiste en la localización del fallo.
- **Identificación de fallos:** caracteriza el fallo por tipo y severidad, estimando instante en que apareció y magnitud.

2.2. Técnicas de diagnóstico

De momento, no existe una taxonomía universalmente aceptada con la que clasificar las diferentes técnicas de diagnóstico. Se encuentran varias técnicas disponibles, pero cabe destacar la realizada por Gertler [Gertler, 1998], que distingue dos técnicas de diagnóstico según el uso de modelos:

- **Técnicas no basadas en modelos:** técnicas de diagnóstico que no emplean modelos matemáticos de los sistemas. Por ejemplo, redundancia física, uso de sensores especiales o razonamiento lógico.

- **Técnicas basadas en modelos:** técnicas de diagnóstico que recurren al uso de modelos matemáticos del sistema. Se distinguen dentro de esta técnica dos grupos, redundancia analítica y estimación de parámetros.

También, podemos destacar la clasificación de sistemas inteligentes en base a diversas característica para diagnosis detallada por Balakrishnan y Honavar [Balakrishnan y Honavar, 1998]:

- **Sistemas Basados en Conocimiento:** técnicas útiles cuando no se tiene o no puede lograrse un modelo analítico del dominio. Estos sistemas tratan de codificar la experiencia de los expertos humanos en forma de principios cualitativos.

- **Sistemas Basados en Casos:** construye un repositorio con casos (ejemplos de diagnosis). Cuando se presenta un nuevo problema de diagnosis, el proceso a seguir es un ciclo de cuatro pasos: *recuperar* de su repositorio el caso o casos más similares al actual. *Reutilizar* la solución adaptándola a la nueva situación. *Revisar* la solución propuesta para verificar su correcta adaptación al problema. Por último, *almacenar* el nuevo caso y su solución.

- **Sistemas de Aprendizaje:** disponen de ejemplos de dominio y además, son capaces de inferir relaciones entre los síntomas y su diagnosis.

- **Sistemas Basados en Modelos:** realizan el razonamiento a partir de modelos de funcionamiento de los sistemas. El proceso de diagnóstico en estos sistemas consiste en detectar las desviaciones que se producen entre el comportamiento que se observa en el sistema y el comportamiento que debería tener según los modelos.

2.3. Diagnóstico basado en consistencia con posibles conflictos

Dentro de los Sistemas Basados en Modelos anteriormente descritos, se encuentra la Diagnosis Basada en Modelos (DBM). La DBM es una técnica de diagnóstico que utiliza modelos matemáticos del comportamiento de los sistemas para poder emular el comportamiento real esperado de un cierto sistema y detectar anomalías del sistema real que representa.

El Diagnóstico basado en consistencia, DBC, es una técnica perteneciente a la diagnosis basada en modelos. A su vez, del DBC deriva el Diagnóstico basado en

consistencia con Posibles Conflictos, que busca definir una diagnosis minimal, entendida como un conjunto de componentes del sistema susceptibles de sufrir un fallo.

Los Posibles Conflictos, PCs, son subsistemas capaces de dar lugar a conflictos en el marco del DBC, es decir, subconjuntos minimales de ecuaciones que contienen la suficiente redundancia analítica como para llevar a cabo diagnóstico de fallos. Los PCs se calculan fuera de línea mediante dos conceptos esenciales:

- Las cadenas evaluables minimales, CEMs, son el conjunto de subsistemas sobredeterminados minimales. Representan una condición necesaria para la existencia de un conflicto.

- Los modelos evaluables minimales, MEMs, son caminos de propagación local, que describen como utilizar las relaciones dentro de una CEM para predecir comportamiento y proporcionar redundancia.

Los PCs se definen como el conjunto de relaciones en una CEM que tiene, como mínimo, un MEM (un camino de propagación local).

Cada MEM detalla un modelo ejecutable que puede emplearse para llevar a cabo detección de fallos. Si entre las predicciones de estos modelos y las observaciones se hallase una discrepancia, el PC es responsable de dicha discrepancia y debería ser confirmado como un conflicto real.

En el cálculo de los PCs, el nodo discrepancia en un MEM se puede hallar de dos formas distintas: como una estimación de una magnitud medida o como una doble estimación de una magnitud no medida.

Un MEM es un conjunto minimal de restricciones necesarias para calcular una discrepancia. No se permite ningún otro nodo discrepancia, en ese caso, no sería minimal.

2.4. Aplicaciones similares

Se ha realizado un estudio de aplicaciones similares y Simbloq [Bregón, 2007] es la única que se ha encontrado. Simbloq es un sistema de simulación de experimentos en bloque que utiliza modelos matemáticos escritos en lenguaje EcosimPro. Los resultados se almacenan en una base de datos y el intercambio de información se efectúa mediante ficheros XML. El sistema es capaz de trabajar independientemente del modelo.

El sistema está formado por cuatro módulos bien diferenciados:

- La parte más importante es Simbloq. Esta parte interactúa con el resto del sistema y dirige la comunicación con el usuario. Es ésta la que permite configurar y simular los diferentes experimentos y modos de fallo que pueden producirse en el sistema.

- La parte llamada Simulación EcosimPro es la perteneciente al entorno de simulación EcosimPro. Se encarga de abstraer el modelo de un sistema y de realizar una simulación de acuerdo a los parámetros seleccionados en

Simbloq, que se comunican mediante una DLL asociada a un experimento de EcosimPro.

- ODBC realiza las tareas de comunicación entra Simbloq y la base de datos.

- Por último, la base de datos almacena los resultados de las simulaciones y la información no incluida en el modelo, como los parámetros y las condiciones de cada simulación.

2.4.1. Comparativa

Tras haber estudiado las características de Simbloq se puede contemplar que la aplicación que se ha realizado en este TFG tiene una ventaja frente a Simbloq. Esta ventaja es que utiliza Matlab en vez de EcosimPro, ya que Matlab es más potente que EcosimPro y es mucho más usado, con lo que cuenta con mucha más información y más librerías disponibles.

Por otro lado, Simbloq es capaz de realizar varias simulaciones en bloque, o lo que es lo mismo, manda realizar varias simulaciones a la vez y se van ejecutando una tras otra, además de ir recogiendo los resultados de las simulaciones. Mientras que nuestra aplicación solo es capaz de realizar una simulación y guardar los resultados. Esto podría ser un trabajo futuro a tener en cuenta.

Capítulo 3
Plan de proyecto

En este capítulo se va a describir la metodología seguida para desarrollar la aplicación, el software utilizado y se va a explicar qué es Java SE. También, se plasmarán las estimaciones, la planificación temporal y el presupuesto.

3.1.Metodología

En el desarrollo de la aplicación se ha seguido el modelo incremental, debido a que se ha ido agregando funcionalidad poco a poco.

El modelo incremental aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. En cada secuencia lineal se va creando el software añadiendo componentes funcionales al sistema, llamados incrementos. Los primeros incrementos son versiones incompletas del producto final.

Este modelo permite una implementación con ampliaciones y/o mejoras progresivas, es decir, en cada paso sucesivo, se actualiza el sistema con nuevas funcionalidades o requisitos. El sistema software se ve como una combinación de resultados sucesivos obtenidos después de cada iteración. Este modelo consta de las siguientes fases:

- Análisis.
- Diseño.
- Desarrollo.
- Pruebas.

3.2. Software utilizado

El software utilizado para la creación de la aplicación es:

- Windows 7: sistema operativo sobre el que se ha trabajado.
- NetBeans IDE: entorno integrado de desarrollo utilizado para programar en lenguaje Java.
- Matlab: software matemático utilizado para la ejecución de los experimentos de simulación y diagnosis.
- StarUML: software utilizado para la creación de los diagramas de análisis y diseño.
- OpenProj: utilizado para la creación del diagrama de Gantt.
- Evolus Pencil: utilizado para el diseño de la interfaz.
- Bloc de notas: editor de texto utilizado para la creación de ficheros *.txt*, necesarios en la ejecución de la aplicación.
- Microsoft Office Word: software utilizado para la creación de esta memoria.

3.3. Java SE

Java es un lenguaje de programación de propósito general orientado a objetos. Ofrece crear aplicaciones independientes de la plataforma, esto quiere decir que los desarrolladores implementan una aplicación una sola vez, válida para cualquier dispositivo, en vez de tener que crear una aplicación para cada uno de ellos.

La plataforma Java Standard Edition o más conocido como Java SE es una colección de APIs del lenguaje de programación Java. Esta plataforma permite desarrollar y desplegar aplicaciones Java en equipos de sobremesa y servidores, así como en los exigentes entornos embebidos. Java SE está compuesta por Java Development Kit (JDK) y Java Runtime Environment (JRE).

JDK es la Plataforma de Desarrollo Java que provee, además del JRE, de herramientas de desarrollo para la creación de programas en Java, como un compilador, depurador, ensamblador, documentador, herramientas de seguridad y de despliegue de aplicaciones, etc. Las utilidades más importantes que se incluyen son:

- javac: es el compilador de Java.
- java: es el intérprete de Java.
- javadoc: genera la documentación de las clases Java de un programa.
- jdb: es el depurador de Java.
- appletviewer: es un visor de applet para generar sus vistas previas, ya que un applet carece de método main y no se puede ejecutar con el programa Java.
- jar: es una herramienta utilizada para manipular los ficheros .jar.
- javah: es un fichero de cabecera para escribir métodos nativos.
- javap: es el desensamblador de Java, que se utiliza para descompilar ficheros compilados.
- extcheck: es una herramienta para detectar conflictos Jar.

JRE es el entorno en tiempo de ejecución de Java. Contiene un conjunto de utilidades que permite la ejecución de programas Java, como una Máquina Virtual de Java (JVM), un conjunto de bibliotecas Java y otros componentes necesarios para que una aplicación Java pueda ser ejecutada. JRE actúa como un intermediario entre el sistema operativo y Java.

JVM es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (bytecode), el cual es generado por el compilador del lenguaje Java.

Por todas estas características del lenguaje de programación Java y de Java SE, se ha decidido desarrollar la aplicación con el lenguaje Java. Además, de poseer conocimientos previos de este lenguaje y de la existencia de la API matlabcontrol, que facilita la comunicación entre Java y Matlab.

3.4.Estimaciones

El presupuesto se ha realizado utilizando la estimación por puntos de función (PF) y la estimación mediante COCOMO.

3.4.1. Puntos de función

La **estimación por puntos de función (PF)** es la siguiente:

- Los valores de los dominios de información y su complejidad se definen de la forma siguiente:

    a. Número de **entradas de usuario** (se cuenta cada entrada de usuario que proporciona diferentes datos orientados a la aplicación):

        - Datos de configuración: complejidad simple.
        - Datos de los valores de entrada: complejidad media.
        - Datos del modelo del sistema: complejidad media.
        - Datos de las variables de entrada: complejidad media.
        - Datos de los parámetros: complejidad media.
        - Datos de las condiciones iniciales: complejidad media.
        - Datos de los fallos paramétricos: complejidad media.
        - Datos de los modelos de simulación: complejidad media.
        - Datos del resultado del cálculo de PCs: complejidad media.
        - Datos de los resultados del experimento de simulación: complejidad media.
        - Datos de la forma de calcular residuos: complejidad simple.
        - Datos de la definición de PCs: complejidad media.
        - Datos de la forma de definir los valores umbral de cada PC: complejidad simple.
        - Datos del algoritmo de identificación de fallos: complejidad media.

    b. Número de **salidas de usuario** (se cuenta cada salida que proporciona al usuario información orientada a la aplicación, informes, pantallas, mensajes de error, etc.):

        - Resultados de experimento de simulación: complejidad simple.

- Resultados en gráfica de experimento de simulación: complejidad simple.
- Resultados de experimento de diagnosis: complejidad simple.
- Resultados en gráfica de experimento de diagnosis: complejidad simple.
- Resultados de aislamiento de fallos: complejidad simple.

c. Número de **consultas de usuario** (se cuenta cada entrada interactiva que genera alguna respuesta del software inmediata en forma de salida interactiva):

d. Número de **ficheros internos**:

e. Número de **ficheros externos**:

- Fichero de resultados de experimento de simulación: complejidad media.
- Fichero de resultados de experimento de diagnosis: complejidad media.
- Manual de usuario: complejidad simple.

- Obtenemos los puntos de función no ajustados (PFNA) mediante una suma ponderada de esas cantidades con los pesos que aparecen a continuación:

Tipo de función	Complejidad	Total x Complejidad	Total por tipo	Suma
Ficheros internos	Simple Media Alta	0 x 7 0 x 10 0 x 15	0	0
Ficheros externos	Simple Media Alta	1 x 5 2 x 7 0 x 10	19	19
Entradas de usuario	Simple Media Alta	3 x 3 11 x 4 0 x 6	53	53
Salidas de usuario	Simple Media Alta	5 x 4 0 x 5 0 x 7	20	12
Consultas de usuario	Simple Media Alta	0 x 3 0 x 4 0 x 6	0	0
Total de puntos de función				**84**

Tabla 1. Puntos de función no ajustados

- Una vez obtenidos los PFNA ajustamos mediante un factor de ajuste (FA). El cálculo del factor de ajuste está basado en 14 características generales de los sistemas que miden la funcionalidad general y complejidad/influencia de la aplicación. A cada característica se le atribuye un peso de 0 a 5 e indica el grado de complejidad/influencia que tiene cada característica.

Grado	Descripción Complejidad	Grado	Descripción Influencia
0	No está presente o su complejidad no es tenida en cuenta	**0**	No está presente o no influye
1	Complejidad mínima	**1**	Influencia mínima
2	Complejidad moderada	**2**	Influencia moderada
3	Complejidad promedio	**3**	Influencia promedio
4	Complejidad significativa	**4**	Influencia significativa
5	Complejidad fuerte	**5**	Influencia fuerte

Tabla 2. Grado de complejidad

- Calculamos el grado de complejidad de cada característica para el cálculo del factor de ajuste:

Factores de ajuste	Complejidad
1. Comunicación de datos	4
2. Funciones distribuidas	0
3. Rendimiento	4
4. Gran carga de trabajo	3
5. Frecuencia de transiciones	2
6. Entrada on-line de datos	0
7. Requisito de manejo del usuario final	2
8. Actualizaciones on-line	0
9. Procesos complejos	3
10. Utilización de otros sistemas	5
11. Facilidad de mantenimiento	1
12. Facilidad de operación	2
13. Instalación en múltiples lugares	1
14. Facilidad de cambio	2
TOTAL:	**29**

Tabla 3. Factores de ajuste

- Cálculo del factor de ajuste (FA) a partir de la suma de los 14 factores de complejidad.

  $FA = (0{,}01 \times \Sigma FC) + 0{,}65 = (0{,}01 \times 29) + 0{,}65 = 0{,}94$

- Cálculo de puntos de función (PF) y obtención del número de líneas de código (LDC) estimadas tomando como referencia la equivalencia en LDC de cada punto de función (53 LDC/PF en Java).

  $PF = PFNA \times FA = 84 \times 0{,}94 = 78{,}96$

  $78{,}96\ PF \times 53\ LDC/PF = 4184{,}88\ LDC \approx 4{,}2\ KLDC$

3.4.2. COCOMO

La **estimación mediante COCOMO** es la siguiente:

$$\text{Esfuerzo nominal} = 2{,}8 \times 4{,}2^{1{,}2} = 15{,}67 \text{ personas-mes}$$

Esfuerzo = 15,67 x 1,15 (fiabilidad) x 1,15 (complejidad) x 0,86 (analistas) x 0,95 (lenguaje) x 0,70 (programadores) x 0,91 (técnicas) = 10,79 personas-mes

Tiempo = 2,5 x $(10,79)^{0,32}$ = 5,35 meses

Nº medio de personas = 10,79/5,35 = 2,02 personas

Para hallar el esfuerzo hemos tenido en cuenta los factores, fiabilidad requerida que es alto, complejidad del software que es alto, calidad de los analistas que es alto, experiencia con el lenguaje de programación utilizado que es alto, calidad de los programadores que es muy alto y técnicas modernas de programación que es alto. Los valores de estos factores los hemos mirado en la Tabla 4.

	Valor de los factores					
FACTORES	Muy bajo	Bajo	Medio	Alto	Muy alto	Extra
Fiabilidad requerida	0,75	0,88	1,00	1,15	1,4	
Tamaño de la base de datos		0,94	1,00	1,08	1,16	
Complejidad del software	0,70	0,85	1,00	1,15	1,30	1,65
Restricciones de tiempo de ejecución			1,00	1,11	1,30	1,66
Restricciones de memoria			1,00	1,06	1,21	1,56
Volatilidad del hardware		0,87	1,00	1,15	1,30	
Restricciones de tiempo de respuesta		0,87	1,00	1,07		
Calidad de los analistas	1,46	1,19	1,00	0,86	0,71	
Experiencia con el tipo de aplicación	1,29	1,13	1,00	0,91	0,82	
Experiencia con el hardware	1,21	1,10	1,00	0,90		
Exp. con el lenguaje de programación	1,14	1,07	1,00	0,95		
Calidad de los programadores	1,42	1,17	1,00	0,86	0,70	
Técnicas modernas de programación	1,24	1,10	1,00	0,91	0,82	
Empleo de herramientas	1,24	1,10	1,00	0,91	0,83	
Restricciones a la duración del proyec.	1,23	1,08	1,00	1,04	1,10	

Tabla 4. Factores de esfuerzo

Los datos anteriores nos indican que el proyecto se llevará a cabo durante 6 meses realizándolo 2 personas.

3.5. Planificación temporal

Las actividades que se deben llevar a cabo, dentro de la planificación temporal establecida anteriormente con las estimaciones, se reflejan en el diagrama de Gantt de la Ilustración 1.

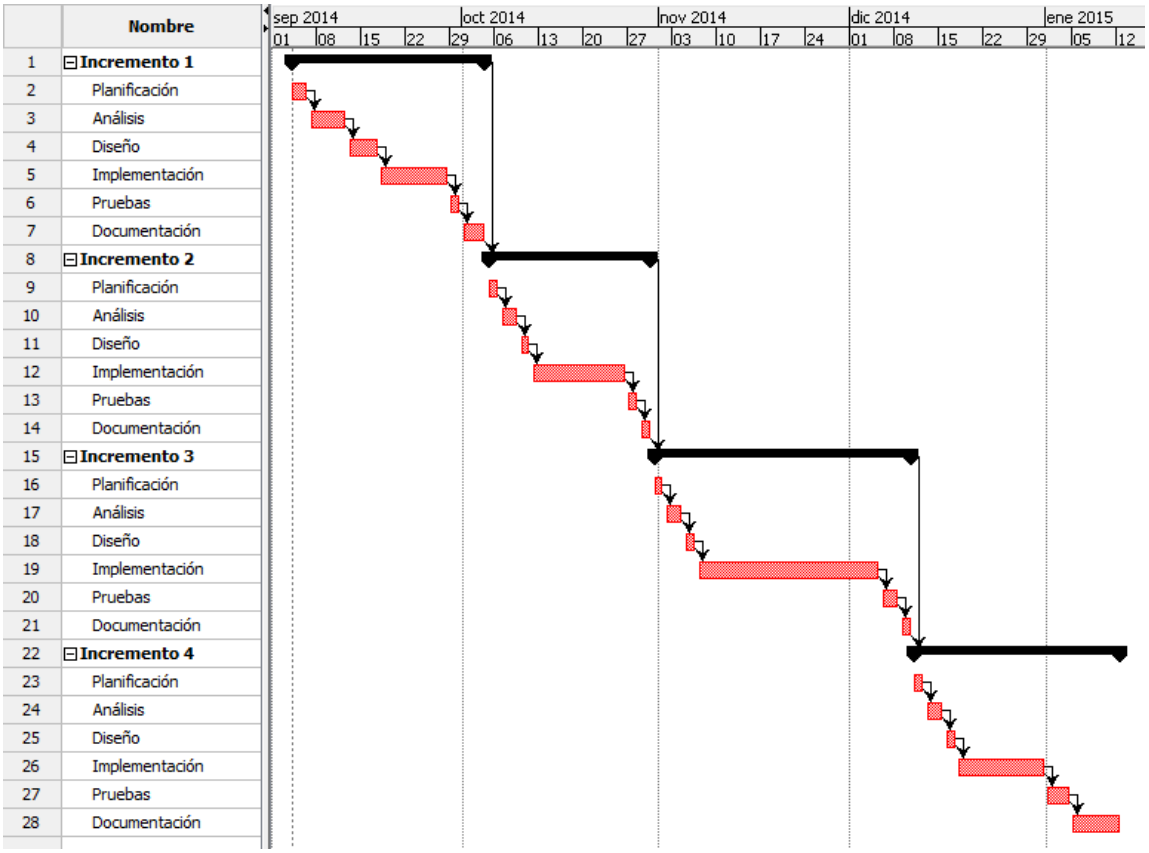


Ilustración 1. Planificación temporal

3.6. Presupuesto

En el presupuesto hay que introducir el coste proporcional al uso que se le darán a los medios Hardware y Software necesarios para desarrollar la aplicación. Naturalmente, también hay que incluir el coste de los recursos humanos utilizados.

3.6.1. Presupuesto inicial

Para realizar el presupuesto inicial utilizaremos los datos obtenidos anteriormente en las estimaciones por puntos de función y de COCOMO.

3.6.1.1. Presupuesto hardware

Para el desarrollo de la aplicación serán necesarios:

- Dos ordenadores personales para el análisis, el desarrollo del sistema, la implantación y pruebas del mismo y la generación de la documentación.
- Conexión a Internet para la obtención de información y la descarga de software.
- Impresora para imprimir la documentación.

HARDWARE	USO (%)	COSTE TOTAL (€)	COSTE (€)
Ordenador personal	12,5	2 x 800	200
Conexión a Internet	(6 meses)	50	300
Impresora	10	150	15
		TOTAL:	**515 €**

Tabla 5. Presupuesto hardware inicial

3.6.1.2. Presupuesto software

Para el desarrollo de la aplicación se utilizarán las siguientes herramientas:

- Windows 7.
- NetBeans IDE.
- Matlab.
- StarUML.
- OpenProj.
- Evolus Pencil.
- Bloc de notas.
- Microsoft Office 2007.

SOFTWARE	USO (%)	COSTE TOTAL (€)	COSTE (€)
Windows 7	12,5	2 x 89,95	22,5
NetBeans IDE	12,5	2 x 0	0
Matlab	12,5	3000	375
StarUML	12,5	2 x 0	0
OpenProj	12,5	2 x 0	0

Evolus Pencil	12,5	2 x 0	0
Bloc de notas	12,5	2 x 0	0
Microsoft Office 2010	(6 meses)	2 x 12,8	153,6

TOTAL: 551,1 €

Tabla 6. Presupuesto software inicial

3.6.1.3. Presupuesto de desarrollo

Teniendo en cuenta que se ha estimado que los trabajos realizados por 2 personas durarán 6 meses, que cada día se pretende trabajar 8 horas (solo días laborables, de lunes a viernes, festivos no incluidos) y que el número de días trabajados al mes son 22, calculamos el número de horas que va a trabajar una persona:

HORAS = 6 x 8 x 22 = 1056 horas realizando el proyecto una persona

	TIEMPO	**COSTE**
Ingeniero	1056 horas	8 €/h
	Total (1 persona):	8448 €

TOTAL (2 personas): **16896 €**

Tabla 7. Presupuesto desarrollo inicial

3.6.1.4. Presupuesto total

La estimación del presupuesto total es la suma de los presupuestos que hemos estimado anteriormente.

PRESUPUESTO	**COSTE**
Hardware	515 €
Software	551,1 €
Desarrollo	16896 €

TOTAL: 17962,1 €

Tabla 8. Presupuesto total inicial

3.6.2. Presupuesto final

Para la realización del presupuesto final se ha tenido en cuenta la duración real del TFG y que ha sido realizado por una sola persona.

3.6.2.1. Presupuesto hardware

En este caso, al ser menor la duración real que la estimada, el coste del hardware es menor y se vuelve a calcular.

HARDWARE	USO (%)	COSTE TOTAL (€)	COSTE (€)
Ordenador personal	10,41	800	83,28
Conexión a Internet	(5 meses)	50	250
Impresora	8,33	150	12,5
		TOTAL:	**345,78 €**

Tabla 9. Presupuesto hardware final

3.6.2.2. Presupuesto software

El presupuesto software se vuelve a calcular ya que la duración real es menor que la estimada.

SOFTWARE	USO (%)	COSTE TOTAL (€)	COSTE (€)
Windows 7	10,41	89,95	9,36
NetBeans IDE	10,41	0	0
Matlab	10,41	3000	321,3
StarUML	10,41	0	0
OpenProj	10,41	0	0
Evolus Pencil	10,41	0	0
Bloc de notas	10,41	0	0
Microsoft Office 2010	(5 meses)	12,8	64
		TOTAL:	**394,66 €**

Tabla 10. Presupuesto software final

3.6.2.3. Presupuesto de desarrollo

El presupuesto de desarrollo se vuelve a calcular ya que el TFG se ha realizado por una sola persona en cuatro meses y medio. Se ha trabajado unos 126 días y una media de 8 horas al día.

HORAS = 126 x 8 = 1008 horas

	TIEMPO	COSTE
Ingeniero	1008 horas	8 €/h
	TOTAL:	**8064 €**

Tabla 11. Presupuesto de desarrollo final

3.6.2.4. Presupuesto total

Se suman todas las cantidades de los presupuestos anteriores para calcular el presupuesto total.

PRESUPUESTO	COSTE
Hardware	345,78 €
Software	394,66 €
Desarrollo	8064 €
TOTAL:	**8804,44 €**

Tabla 12. Presupuesto total final

3.6.3. Conclusiones

En un principio se había estimado la necesidad de dos personas para la realización del trabajo durante seis meses y al final se ha realizado por una persona en cuatro meses y medio. Por esto, el presupuesto final es mucho menor que el inicial, ya que el mayor coste es el de desarrollo.

Capítulo 4
Análisis

En esta sección, se va a definir tanto la funcionalidad del sistema, como su interacción con los usuarios y/u otros sistemas. Todo esto se llevará a cabo mediante la identificación de los usuarios, definición de los requisitos, diagrama de casos de uso, diagrama de clases y diagramas de secuencia.

4.1.Identificación de los actores

Con este sistema van a interactuar dos actores, el usuario final de la aplicación y el sistema Matlab.

ACT01	*Usuario*
Versión	*1.0*
Descripción	Este actor representa a la persona que va a interactuar con el sistema para configurar y ejecutar experimentos de simulación y diagnosis.
Comentarios	*Ninguno*

Tabla 13. ACT01 - Usuario

ACT02	*Matlab*
Versión	*1.0*
Descripción	Este actor representa al sistema Matlab, el cual, va a interactuar con el sistema para ejecutar y devolverle los resultados de los experimentos de simulación y diagnosis.
Comentarios	*Ninguno*

Tabla 14. ACT02 - Matlab

4.2.Requisitos

4.2.1. Requisitos funcionales

Los requisitos funcionales definen la funcionalidad del sistema o sus componentes. En este caso son:

- RQF01 Cargar los valores de entrada.
- RQF02 Cargar el modelo del sistema.
- RQF03 Cargar las variables de entrada.
- RQF04 Cargar los parámetros.
- RQF05 Cargar las condiciones iniciales.
- RQF06 Cargar los fallos paramétricos.
- RQF07 Configurar experimento.
- RQF08 Ejecutar experimento de simulación.
- RQF09 Mostrar resultados del experimento de simulación.
- RQF10 Mostrar gráficamente los resultados del experimento de simulación.
- RQF11 Guardar resultados del experimento de simulación.
- RQF12 Cargar los modelos de simulación.

- RQF13 Cargar el resultado del cálculo de PCs.
- RQF14 Cargar los resultados de un experimento de simulación.
- RQF15 Ejecutar experimento de diagnosis.
- RQF16 Elegir forma de calcular los residuos.
- RQF17 Calcular los residuos.
- RQF18 Mostrar resultados del experimento de diagnosis.
- RQF19 Mostrar gráficamente los resultados del experimento de diagnosis.
- RQF20 Cargar la definición de PCs.
- RQF21 Elegir forma de definir los valores umbral de cada PC.
- RQF22 Aislar los fallos.
- RQF23 Guardar resultados del experimento de diagnosis.
- RQF24 Cargar algoritmo de identificación de fallos.
- RQF25 Identificar los fallos.

RQF01	*Cargar los valores de entrada*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario cargar el fichero que contenga los valores de entrada de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 15. RQF01 - Cargar los valores de entrada

RQF02	*Cargar el modelo del sistema*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario cargar el fichero que contenga el modelo de un sistema.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*

Comentarios	*Ninguno*

Tabla 16. RQF02 - Cargar el modelo del sistema

RQF03	*Cargar las variables de entrada*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario cargar el fichero que contenga las variables de entrada de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 17. RQF03 - Cargar las variables de entrada

RQF04	*Cargar los parámetros*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario cargar el fichero que contenga los parámetros de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 18. RQF04 - Cargar los parámetros

RQF05	*Cargar las condiciones iniciales*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario cargar el fichero que contenga las condiciones iniciales de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 19. RQF05 - Cragar las condiciones iniciales

RQF06	*Cargar los fallos paramétricos*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario cargar el fichero que contenga los fallos paramétricos de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 20. RQF06 - Cargar los fallos paramétricos

RQF07	*Configurar Experimento*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario modificar los valores de las variables de configuración de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 21. RQF07 - Configurar experimento

RQF08	*Ejecutar experimento de simulación*
Versión	*1.0*
Dependencias	- RQF01 Cargar los valores de entrada. - RQF02 Cargar el modelo del sistema. - RQF03 Cargar las variables de entrada. - RQF04 Cargar los parámetros. - RQF05 Cargar las condiciones iniciales. - RQF06 Cargar los fallos paramétricos. - RQF07 Configurar experimento.
Descripción	El sistema deberá conectar con Matlab para ejecutar un experimento de simulación.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 22. RQF08 - Ejecutar experimento de simulación

RQF09	*Mostrar resultados del experimento de simulación*
Versión	*1.0*
Dependencias	- RQF08 Ejecutar experimento de simulación.
Descripción	El sistema deberá mostrar al usuario los resultados del experimento de simulación.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*El usuario prefiere que se muestren en una tabla.*

Tabla 23. RQF09 - Mostrar resultados del experimento de simulación

RQF10	*Mostrar gráficamente los resultados del experimento de simulación*
Versión	*1.0*
Dependencias	- RQF08 Ejecutar experimento de simulación.
Descripción	El sistema deberá mostrar gráficamente al usuario los resultados del experimento de simulación.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 24. RQF10 - Mostrar gráficamente resultados del experimento de simulación

RQF11	*Guardar resultados del experimento de simulación*
Versión	*1.0*
Dependencias	- RQF08 Ejecutar experimento de simulación.
Descripción	El sistema deberá permitir al usuario guardar en un fichero los resultados del experimento de simulación.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 25. RQF11 - Guardar resultados del experimento de simulación

RQF12	*Cargar los modelos de diagnosis*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario cargar los ficheros que contengan los modelos de simulación de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 26. RQF12 - Cargar los modelos de diagnosis

RQF13	*Cargar el resultado del cálculo de PCs*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario cargar el fichero que contenga el resultado del cálculo de PCs de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 27. RQF13 - Cargar el resultado del cálculo de PCs

RQF14	*Cargar los resultados de un experimento de simulación*
Versión	*1.0*
Dependencias	- RQF08 Ejecutar experimento de simulación. - RQF11 Guardar resultados del experimento de simulación.
Descripción	El sistema deberá permitir al usuario cargar el fichero que contenga los resultados de un experimento de simulación.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 28. RQF14 - Cargar los resultados de un experimento de simulación

RQF15	*Ejecutar experimento de diagnosis*
Versión	*1.0*
Dependencias	- RQF12 Cargar los modelos de simulación. - RQF13 Cargar el resultado del cálculo de PCs. - RQF14 Cargar los resultados de un experimento de simulación.
Descripción	El sistema deberá conectar con Matlab para ejecutar un experimento de diagnosis.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 29. RQF15 - Ejecutar experimento de diagnosis

RQF16	*Elegir forma de calcular los residuos*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario elegir la forma de calcular los residuos de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Se permiten 3 formas de calcular residuos:* - *Diferencia.* - *Valor absoluto de la diferencia.* - *Distancia euclídea.*

Tabla 30. RQF16 - Elegir forma de calcular los residuos

RQF17	*Calcular los residuos*
Versión	*1.0*
Dependencias	- RQF16 Elegir forma de calcular los residuos. - RQF15 Ejecutar experimento de diagnosis.
Descripción	El sistema deberá calcular los residuos de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 31. RQF17 - Calcular los residuos

RQF18	*Mostrar resultados del experimento de diagnosis*
Versión	*1.0*
Dependencias	- RQF15 Ejecutar experimento de diagnosis.
Descripción	El sistema deberá mostrar al usuario los resultados del experimento de diagnosis.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*El usuario prefiere que se muestren en una tabla.*

Tabla 32. RQF18 - Mostrar resultados del experimento de diagnosis

RQF19	*Mostrar gráficamente los resultados del experimento de diagnosis*
Versión	*1.0*
Dependencias	- RQF15 Ejecutar experimento de diagnosis.
Descripción	El sistema deberá mostrar gráficamente al usuario los resultados del experimento de diagnosis.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 33. RQF19 - Mostrar gráficamente los resultados del experimento de diagnosis

RQF20	*Cargar la definición de PCs*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario cargar el fichero que contenga la definición de PCs de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 34. RQF20 - Cargar la definición de PCs

RQF21	*Elegir forma de definir los valores umbral de cada PC*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario elegir la forma de definir los valores umbral de cada PC de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Se permiten 3 formas de definir los valores umbral de cada PC:* - *Realizar Z-test.* - *Utilizar el valor global definido en la configuración.* - *Introducir un valor de umbral para cada Posible Conflicto.*

Tabla 35. RQF21 - Elegir forma de definir los valores umbral de cada PC

RQF22	*Aislar los fallos*
Versión	*1.0*
Dependencias	- RQF15 Ejecutar experimento de diagnosis.
Descripción	El sistema deberá aislar los fallos de un experimento.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 36. RQF22 - Aislar los fallos

RQF23	*Guardar resultados del experimento de diagnosis*
Versión	*1.0*
Dependencias	- RQF15 Ejecutar experimento de diagnosis.
Descripción	El sistema deberá permitir al usuario guardar en un fichero los resultados del experimento de diagnosis.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 37. RQF23 - Guardar resultados del experimento de diagnosis

RQF24	*Cargar algoritmo de identificación de fallos*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir al usuario cargar el fichero que contenga el algoritmo de identificación de fallos de un experimento de diagnosis.
Importancia	*Estaría bien*
Prioridad	*Estaría bien*
Comentarios	*Ninguno*

Tabla 38. RQF24 - Cargar algoritmo de identificación de fallos

RQF25	*Identificar los fallos*
Versión	*1.0*
Dependencias	- RQF15 Ejecutar experimento de diagnosis.
Descripción	El sistema deberá identificar los fallos de un experimento.
Importancia	*Estaría bien*
Prioridad	*Estaría bien*
Comentarios	*Ninguno*

Tabla 39. RQF25 - Identificar los fallos

4.2.2. Requisitos no funcionales

Los requisitos no funcionales describen propiedades o cualidades que el sistema debe tener. En este caso son:

- RQNF01 Mantenibilidad.
- RQNF02 Modificabilidad.
- RQNF03 Usabilidad.

RQNF01	*Mantenibilidad*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá ser fácilmente mantenible.
Importancia	*Muy Alta*

Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 40. RQNF01 - Mantenibilidad

RQNF02	*Modificabilidad*
Versión	*1.0*
Dependencias	
Descripción	El sistema deberá permitir añadir nuevos algoritmos.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 41. RQNF02 - Modificabilidad

RQNF03	*Usabilidad*
Versión	*1.0*
Dependencias	
Descripción	El usuario deberá poder utilizar la aplicación sin problemas después de haber leído el manual de usuario.
Importancia	*Muy Alta*
Prioridad	*Muy Alta*
Comentarios	*Ninguno*

Tabla 42. RQNF03 - Usabilidad

4.3. Casos de uso

A partir de los casos de uso podemos especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas.

4.3.1. Diagrama de casos de uso

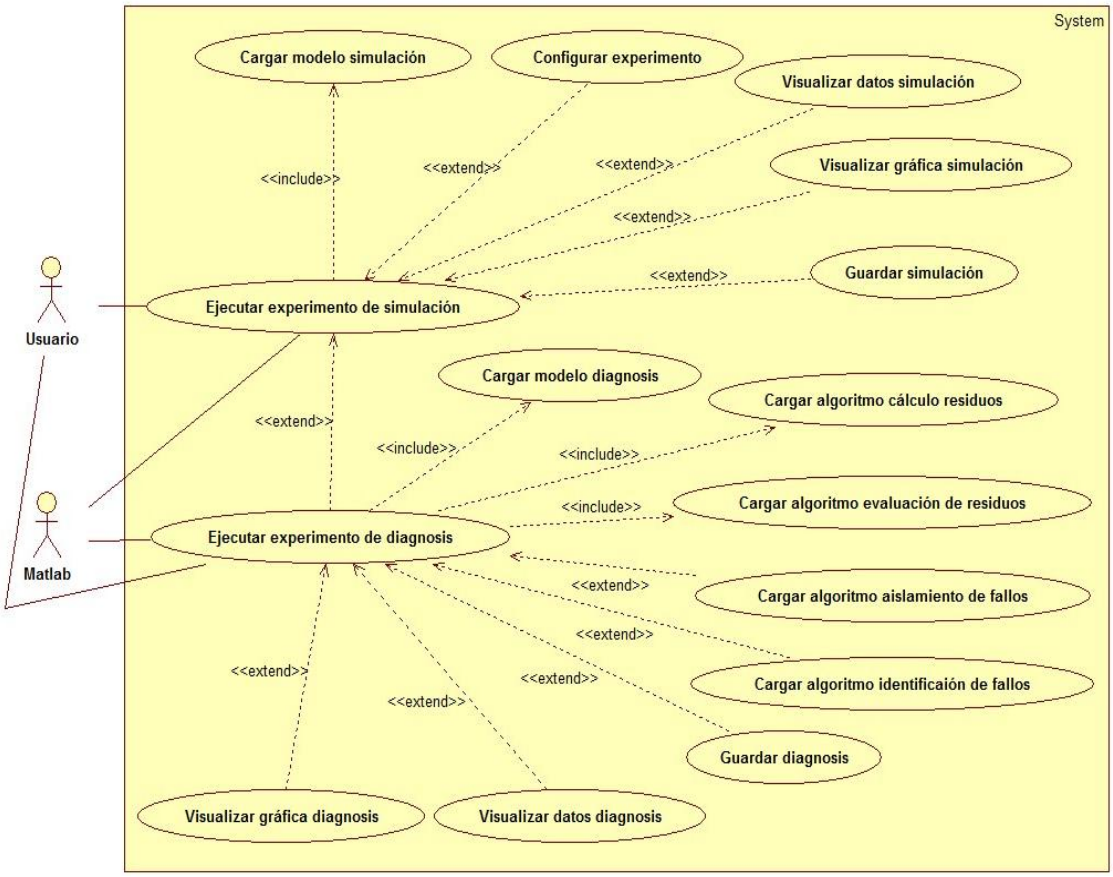


Ilustración 2. Diagrama de casos de uso

4.3.2. Especificación de casos de uso

UC01	*Cargar modelo simulación*
Versión	*1.0*
Dependencias	- RQF01 Cargar los valores de entrada. - RQF02 Cargar el modelo del sistema. - RQF03 Cargar las variables de entrada. - RQF04 Cargar los parámetros. - RQF05 Cargar las condiciones iniciales. - RQF06 Cargar los fallos paramétricos.

Precondición			
Descripción	El sistema deberá permitir al usuario cargar los ficheros necesarios para la ejecución de un experimento de simulación.		
Secuencia Normal	**Paso**	**Acción**	
	1	El usuario pulsa el botón "Cargar Modelo Simulación".	
	2	El sistema abre una ventana de selección de archivos.	
	3	El usuario selecciona los ficheros necesarios.	
		3.a	Selecciona el fichero con los valores de entrada.
		3.b	Selecciona el fichero con el modelo del sistema.
		3.c	Selecciona el fichero con las variables de entrada.
		3.d	Selecciona el fichero con los parámetros.
		3.e	Selecciona el fichero con las condiciones iniciales.
		3.f	Selecciona el fichero con los fallos paramétricos.
	4	El sistema carga los ficheros.	
Postcondición	El sistema ha cargado correctamente el modelo de simulación.		
Excepciones	**Paso**	**Acción**	
	4	El sistema no carga correctamente los ficheros.	
		1	El sistema muestra un mensaje indicando que no se han cargado correctamente los ficheros.
		2	Finaliza el caso de uso.
Importancia	*Alta*		
Prioridad	*Alta*		
Comentarios	*Ninguno*		

Tabla 43. UC01 - Cargar modelo simulación

UC02	*Configurar experimento*	
Versión	*1.0*	
Dependencias	- RQF07 Configurar experimento.	
Precondición		
Descripción	El sistema deberá permitir al usuario modificar los valores de las variables de configuración de un experimento.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario introduce el valor DeltaT, Umbral o Tsim en el campo del formulario correspondiente.
	2	El sistema comprueba que el valor sea un número decimal.
Postcondición	El experimento ha sido configurado.	
Excepciones	**Paso**	**Acción**
	2	El valor DeltaT, Umbral o Tsim no es válido.
	1	El sistema muestra un mensaje indicando que el valor DeltaT, Umbral o Tsim no es válido.
	2	Finaliza el caso de uso.
Importancia	*Alta*	
Prioridad	*Alta*	
Comentarios	*Ninguno*	

Tabla 44. UC02 - Configurar experimento

UC03	*Ejecutar experimento de simulación*	
Versión	*1.0*	
Dependencias	- RQF08 Ejecutar experimento de simulación. - UC01 Cargar modelo de simulación. - UC02 Configurar experimento.	
Precondición	El sistema deberá haber cargado correctamente todos los ficheros necesarios y los datos de configuración del experimento de simulación.	
Descripción	El sistema deberá permitir ejecutar un experimento de simulación mediante Matlab.	
Secuencia Normal	**Paso**	**Acción**

	1	El usuario pulsa el botón "Ejecutar Experimento Simulación".
	2	El sistema facilita a Matlab los datos necesarios para la ejecución de un experimento de simulación.
	3	Matlab ejecuta el experimento de simulación.
	4	Matlab devuelve al sistema los resultados del experimento de simulación.
Postcondición	La ejecución del experimento de simulación se ha realizado correctamente.	
Importancia	*Alta*	
Prioridad	*Alta*	
Comentarios	*Ninguno*	

Tabla 45. UC03 - Ejecutar experimento de simulación

UC04	*Visualizar datos simulación*	
Versión	*1.0*	
Dependencias	- RQF08 Ejecutar experimento de simulación. - RQF09 Mostrar resultados del experimento de simulación. - UC03 Ejecutar experimento de simulación.	
Precondición	El sistema deberá haber obtenido los resultados de un experimento de simulación.	
Descripción	El sistema deberá mostrar al usuario en una tabla los resultados de un experimento de simulación.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa el botón "Ver Datos Simulación".
	2	El sistema muestra al usuario en una tabla los resultados de un experimento de simulación.
Postcondición	El sistema muestra los resultados de un experimento de simulación correctamente.	
Importancia	*Alta*	
Prioridad	*Alta*	
Comentarios	*Ninguno*	

Tabla 46. UC04 - Visualizar datos simulación

UC05	*Visualizar gráfica simulación*	
Versión	*1.0*	
Dependencias	- RQF08 Ejecutar experimento de simulación. - RQF10 Mostrar gráficamente los resultados del experimento de simulación. - UC03 Ejecutar experimento de simulación.	
Precondición	El sistema deberá haber obtenido los resultados de un experimento de simulación.	
Descripción	El sistema deberá mostrar al usuario en una gráfica los resultados de un experimento de simulación.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa el botón "Ver Gráfica Simulación".
	2	El sistema muestra al usuario una ventana con todas las variables del experimento de simulación.
	3	El usuario selecciona las variables que desea que se muestren en la gráfica.
	4	El sistema muestra al usuario en una gráfica los resultados de un experimento de simulación correspondientes a las variables seleccionadas.
Postcondición	El sistema muestra gráficamente los resultados de un experimento de simulación correctamente.	
Excepciones	**Paso**	**Acción**
	3	El usuario no ha seleccionado ninguna variable.
		1 Finaliza el caso de uso.
Importancia	*Alta*	
Prioridad	*Alta*	
Comentarios	*Ninguno*	

Tabla 47. UC05 - Visualizar gráfica simulación

UC06	*Guardar simulación*
Versión	*1.0*
Dependencias	- RQF08 Ejecutar experimento de simulación. - RQF11 Guardar resultados del experimento de simulación. - UC03 Ejecutar experimento de simulación.

Precondición	El sistema deberá haber obtenido los resultados de un experimento de simulación.		
Descripción	El sistema deberá permitir al usuario guardar en un fichero .txt los resultados de un experimento de simulación.		
Secuencia Normal	**Paso**	**Acción**	
	1	El usuario pulsa el botón "Guardar Datos Simulación".	
	2	El sistema abre una ventana de selección de archivos.	
	3	El usuario elige la ubicación y el nombre del fichero.	
	4	El sistema crea el fichero con el nombre y en la ubicación elegida por el usuario.	
	5	El sistema guarda los resultados del experimento de simulación en el fichero anteriormente creado.	
Postcondición	El sistema ha creado correctamente un fichero con los resultados de un experimento de simulación.		
Excepciones	**Paso**	**Acción**	
	3	El usuario no elige la ubicación y ni el nombre del fichero.	
		1	Finaliza el caso de uso.
Importancia	*Alta*		
Prioridad	*Alta*		
Comentarios	*Ninguno*		

Tabla 48. UC06 - Guardar simulación

UC07	*Cargar modelo diagnosis*
Versión	*1.0*
Dependencias	- RQF03 Cargar las variables de entrada. - RQF04 Cargar los parámetros. - RQF05 Cargar las condiciones iniciales. - RQF12 Cargar los modelos de simulación. - RQF13 Cargar el resultado del cálculo de PCs. - RQF14 Cargar los resultados de un experimento de simulación.
Precondición	
Descripción	El sistema deberá permitir al usuario cargar los ficheros necesarios para la ejecución de un experimento de diagnosis.

Secuencia Normal	**Paso**	**Acción**	
	1	El usuario pulsa el botón "Cargar Modelo Diagnosis".	
	2	El sistema abre una ventana de selección de archivos.	
	3	El usuario selecciona los ficheros necesarios.	
	4	El sistema carga los ficheros.	
Postcondición	El sistema ha cargado correctamente el modelo de diagnosis.		
Excepciones	**Paso**	**Acción**	
	4	El sistema no carga correctamente los ficheros.	
		1	El sistema muestra un mensaje indicando que no se han cargado correctamente los ficheros.
		2	Finaliza el caso de uso.
Importancia	*Alta*		
Prioridad	*Alta*		
Comentarios	*Ninguno*		

Tabla 49. UC07 - Cargar modelo diagnosis

UC08	*Ejecutar experimento de diagnosis*		
Versión	*1.0*		
Dependencias	- RQF15 Ejecutar experimento de diagnosis. - UC07 Cargar modelo diagnosis.		
Precondición	El sistema deberá haber cargado correctamente todos los ficheros necesarios del experimento de diagnosis.		
Descripción	El sistema deberá permitir ejecutar un experimento de diagnosis mediante Matlab.		
Secuencia Normal	**Paso**	**Acción**	
	1	El usuario pulsa el botón "Ejecutar Experimento Diagnosis".	
	2	El sistema facilita a Matlab los datos necesarios para la ejecución de un experimento de diagnosis.	
	3	Matlab ejecuta el experimento de diagnosis.	
	4	Matlab devuelve al sistema los resultados del experimento de diagnosis.	

Postcondición	La ejecución del experimento de diagnosis se ha realizado correctamente.
Importancia	*Alta*
Prioridad	*Alta*
Comentarios	*Ninguno*

Tabla 50. UC08 - Ejecutar experimento de diagnosis

UC09	*Cargar algoritmo cálculo de residuos*	
Versión	*1.0*	
Dependencias	- RQF16 Elegir forma de calcular los residuos.	
Precondición		
Descripción	El sistema deberá permitir al usuario elegir el algoritmo con que se efectúe el cálculo de residuos.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa el botón "Elegir Residuos".
	2	El sistema abre una ventana con un desplegable con los algoritmos de cálculo de residuos.
	3	El usuario selecciona un algoritmo.
	4	El sistema carga el algoritmo.
Postcondición	El sistema ha cargado correctamente el algoritmo.	
Excepciones	**Paso**	**Acción**
	3	El usuario no selecciona un algoritmo.
		1 Finaliza el caso de uso.
Importancia	*Alta*	
Prioridad	*Alta*	
Comentarios	*Ninguno*	

Tabla 51. UC09 - Cargar algoritmo cálculo de residuos

UC10	*Cargar algoritmo evaluación de residuos*
Versión	*1.0*
Dependencias	- RQF16 Elegir forma de calcular los residuos. - RQF17 Calcular los residuos. - RQF15 Ejecutar experimento de diagnosis.

	- UC08 Ejecutar experimento de diagnosis. - UC09 Cargar algoritmo cálculo de residuos.	
Precondición	El sistema deberá haber obtenido los resultados de un experimento de diagnosis y el usuario deberá haber elegido el algoritmo para el cálculo de residuos.	
Descripción	El sistema deberá calcular los residuos de un experimento de diagnosis.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa el botón "Evaluar Residuos".
	2	El sistema calcula los residuos.
Postcondición	El sistema ha calculado correctamente los residuos de un experimento de diagnosis.	
Importancia	*Alta*	
Prioridad	*Alta*	
Comentarios	*Ninguno*	

Tabla 52. UC10 - Cargar algoritmo evaluación de residuos

UC11	*Cargar algoritmo aislamiento de fallos*	
Versión	*1.0*	
Dependencias	- RQF15 Ejecutar experimento de diagnosis. - RQF20 Cargar la definición de PCs. - RQF21 Elegir forma de definir los valores umbral de cada PC. - RQF22 Aislar los fallos. - UC08 Ejecutar experimento de diagnosis.	
Precondición	El sistema deberá haber obtenido los resultados de un experimento de diagnosis.	
Descripción	EL sistema deberá permitir al usuario cargar la definición de PCs y elegir la forma de definir los valores umbral de cada PC. Además, de aislar los fallos.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa el botón "Aislar fallos".
	2	El sistema abre una ventana de selección de archivos.
	3	El usuario selecciona el fichero con la definición de PCs.
	4	El sistema carga el fichero.

	5	El sistema abre una ventana con un desplegable con las distintas formas de definir los valores umbral de cada PC.
	6	El usuario selecciona una forma de definir los valores umbral de cada PC.
	7	El sistema carga la forma de definir los valores umbral de cada PC..
	8	El sistema aísla los fallos.
	9	El sistema muestra los resultados.
Postcondición	El sistema ha aislado correctamente los fallos de un experimento de diagnosis.	
Excepciones	**Paso**	**Acción**
	3	El usuario no selecciona un fichero.
		1 Finaliza el caso de uso.
	6	El usuario no selecciona una forma de definir los valores umbral de cada PC.
		1 Finaliza el caso de uso.
Importancia	*Alta*	
Prioridad	*Alta*	
Comentarios	*Ninguno*	

Tabla 53. UC11 - Cargar algoritmo aislamiento de fallos

UC12	*Cargar algoritmo identificación de fallos*	
Versión	*1.0*	
Dependencias	- RQF15 Ejecutar experimento de diagnosis. - RQF24 Cargar algoritmo de identificación de fallos. - RQF25 Identificar los fallos. - UC08 Ejecutar experimento de diagnosis.	
Precondición	El sistema deberá haber obtenido los resultados de un experimento de diagnosis.	
Descripción	El sistema deberá permitir al usuario elegir un algoritmo de identificación de fallos y realizar la identificación de fallos.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa el botón "Identificar fallos".
	2	El sistema abre una ventana de selección de

		archivos.
	3	El usuario selecciona el fichero con el algoritmo de identificación de fallos.
	4	El sistema carga el fichero.
	5	El sistema identifica los fallos.
	6	El sistema muestra los resultados.
Postcondición	El sistema ha identificado correctamente los fallos de un experimento de diagnosis.	
Excepciones	**Paso**	**Acción**
	3	El usuario no selecciona un fichero.
		1 Finaliza el caso de uso.
Importancia	*Alta*	
Prioridad	*Alta*	
Comentarios	*Ninguno*	

Tabla 54. UC12 - Cargar algoritmo identificación de fallos

UC13	*Visualizar datos diagnosis*	
Versión	*1.0*	
Dependencias	- RQF15 Ejecutar experimento de diagnosis. - RQF19 Mostrar resultados del experimento de diagnosis. - UC08 Ejecutar experimento de diagnosis.	
Precondición	El sistema deberá haber obtenido los resultados de un experimento de diagnosis.	
Descripción	El sistema deberá mostrar al usuario en una tabla los resultados de un experimento de diagnosis.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa el botón "Ver Datos Diagnosis".
	2	El sistema muestra al usuario en una tabla los resultados de un experimento de diagnosis.
Postcondición	El sistema muestra los resultados de un experimento de diagnosis correctamente.	
Importancia	*Alta*	
Prioridad	*Alta*	

Comentarios	*Ninguno*

Tabla 55. UC13 - Visualizar datos diagnosis

UC14	*Visualizar gráfica diagnosis*	
Versión	*1.0*	
Dependencias	- RQF15 Ejecutar experimento de diagnosis. - RQF19 Mostrar gráficamente los resultados del experimento de diagnosis. - UC08 Ejecutar experimento de diagnosis.	
Precondición	El sistema deberá haber obtenido los resultados de un experimento de diagnosis.	
Descripción	El sistema deberá mostrar al usuario en una gráfica los resultados de un experimento de diagnosis.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa el botón "Ver Gráfica Diagnosis".
	2	El sistema muestra al usuario una ventana con todos los modelos de simulación del experimento de diagnosis.
	3	El usuario selecciona el modelo de simulación que desea que se muestre en la gráfica.
	4	El sistema muestra al usuario en una gráfica los resultados de un experimento de diagnosis correspondientes a las variables seleccionadas.
Postcondición	El sistema muestra gráficamente los resultados de un experimento de diagnosis correctamente.	
Excepciones	**Paso**	**Acción**
	3	El usuario no ha seleccionado ningún modelo de simulación.
		1 Finaliza el caso de uso.
Importancia	*Alta*	
Prioridad	*Alta*	
Comentarios	*Ninguno*	

Tabla 56. UC14 - Visualizar gráfica diagnosis

UC15	*Guardar diagnosis*	
Versión	*1.0*	
Dependencias	- RQF15 Ejecutar experimento de diagnosis. - RQF23 Guardar resultados del experimento de diagnosis. - UC08 Ejecutar experimento de diagnosis.	
Precondición	El sistema deberá haber obtenido los resultados de un experimento de diagnosis.	
Descripción	El sistema deberá permitir al usuario guardar en un fichero .txt los resultados de un experimento de diagnosis.	
Secuencia Normal	**Paso**	**Secuencia Normal**
	1	El usuario pulsa el botón "Guardar Datos Diagnosis".
	2	El sistema abre una ventana de selección de archivos.
	3	El usuario elige la ubicación y el nombre del fichero.
	4	El sistema crea el fichero con el nombre y en la ubicación elegida por el usuario.
	5	El sistema guarda los resultados del experimento de diagnosis en el fichero anteriormente creado.
Postcondición	El sistema ha creado correctamente un fichero con los resultados de un experimento de diagnosis.	
Excepciones	**Paso**	**Excepciones**
	3	El usuario no elige la ubicación y ni el nombre del fichero.
		1 Finaliza el caso de uso.
Importancia	*Alta*	
Prioridad	*Alta*	
Comentarios	*Ninguno*	

Tabla 57. UC15 - Guardar diagnosis

4.4.Diagrama de clases

En este apartado, la estructura del sistema a desarrollar en este TFG se va a describir mostrando sus clases con ayuda del diagrama de clases.

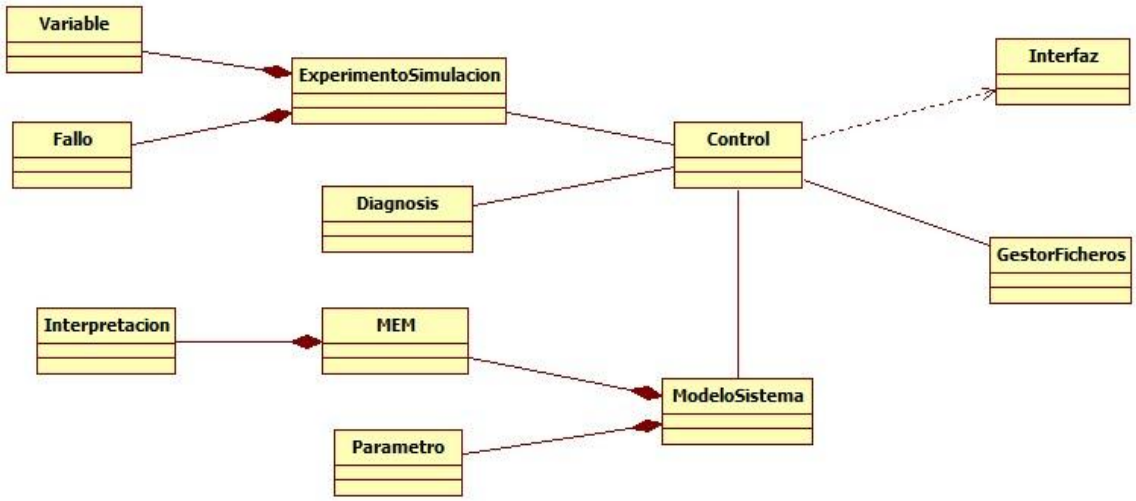


Ilustración 3. Diagrama de clases de análisis

Como podemos ver en la Ilustración 3, nuestro sistema tiene seis clases más importantes, de las que no depende su existencia de ninguna otra. Una de estas clases es la llamada Control que hace de controlador de todo el sistema junto con otra de estas llamada GestorFicheros que se encarga de la gestión de los ficheros con los que trabaja el sistema. La clase denominada Interfaz contiene todas las interfaces del sistema. La clase designada con el nombre ModeloSistema es la que contiene los parámetros y algoritmos necesarios en los experimentos. ExperimentoSimulación es otra de esas clases importantes que contiene las variables y los fallos necesarios en los experimentos de simulación. Igualmente la clase Diagnosis contiene todo lo necesario para los experimentos de diagnosis.

4.5.Diagramas de secuencia

Los diagramas de secuencia muestran la interacción de un conjunto de objetos de un sistema a través del tiempo. Además, contienen mensajes intercambiados entre los objetos y detalles de implementación del escenario, incluyendo objetos y clases. Por cada caso de uso se modela un diagrama.

En este documento se muestran los diagramas de secuencia más relevantes. En el CD-ROM que se entrega junto a este documento se encuentran todos los diagramas en la carpeta "Diagramas de secuencia".

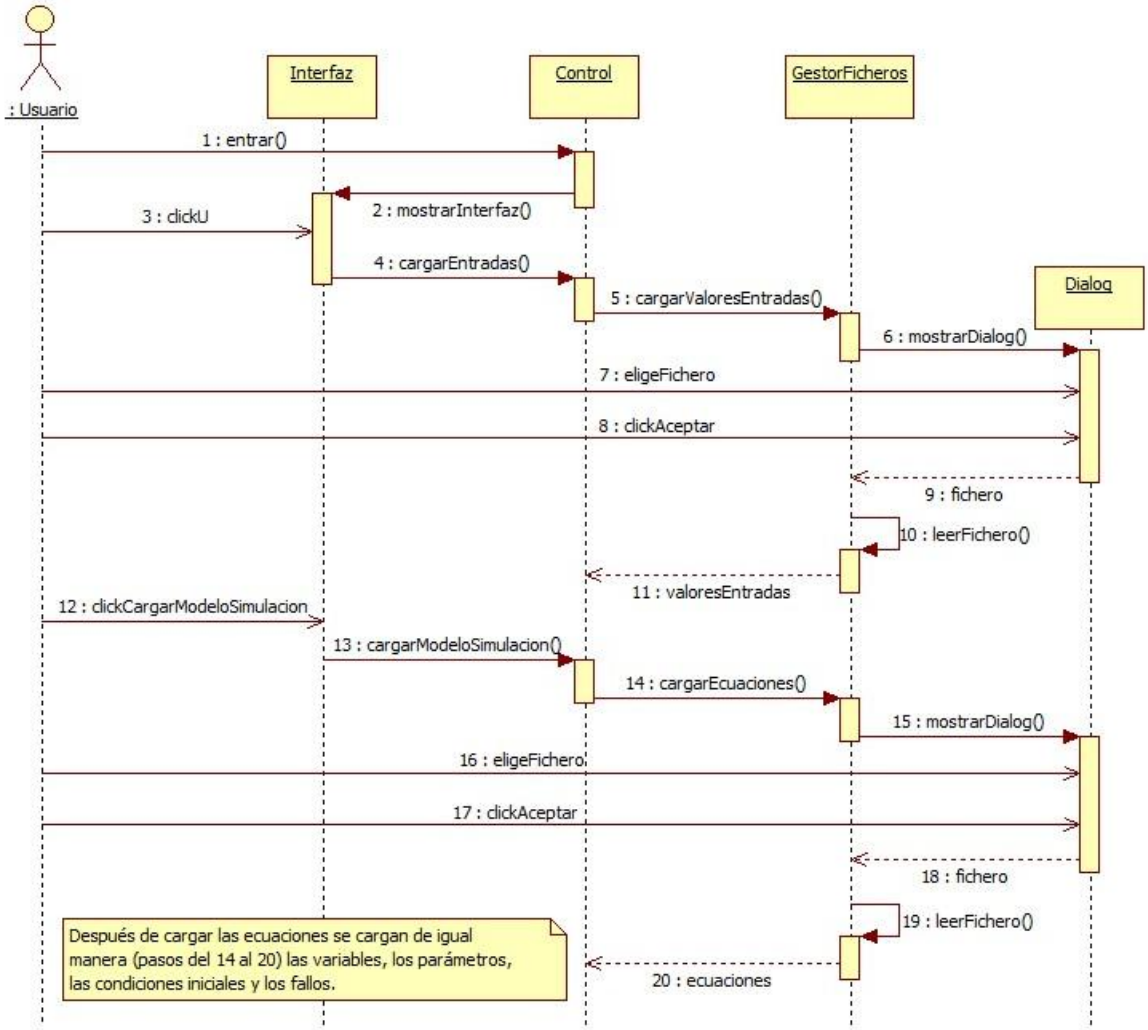


Ilustración 4. Diagrama de secuencia "Cargar modelo simulación"

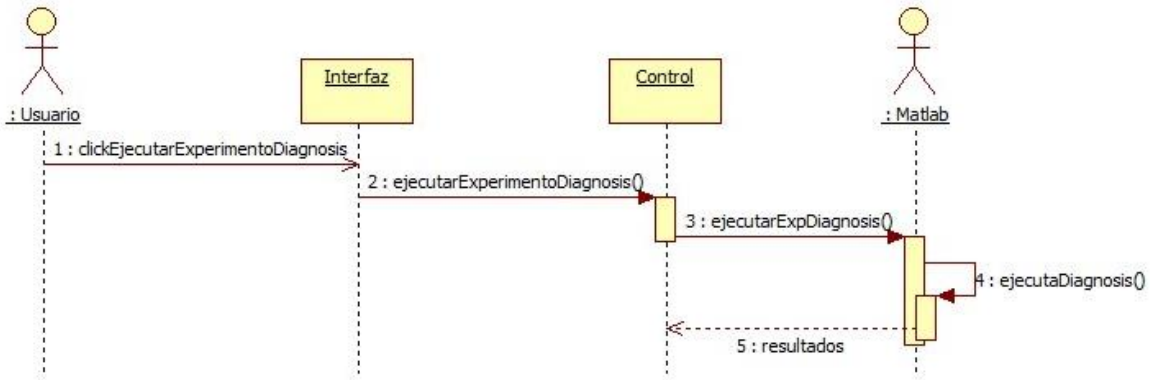


Ilustración 5. Diagrama de secuencia "Ejecutar experimento de diagnosis"

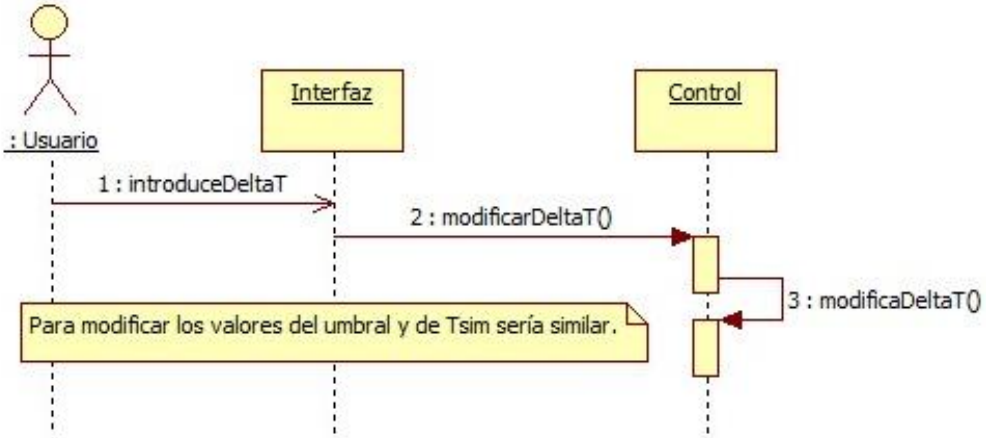


Ilustración 6. Diagrama de secuencia "Configurar experimento"

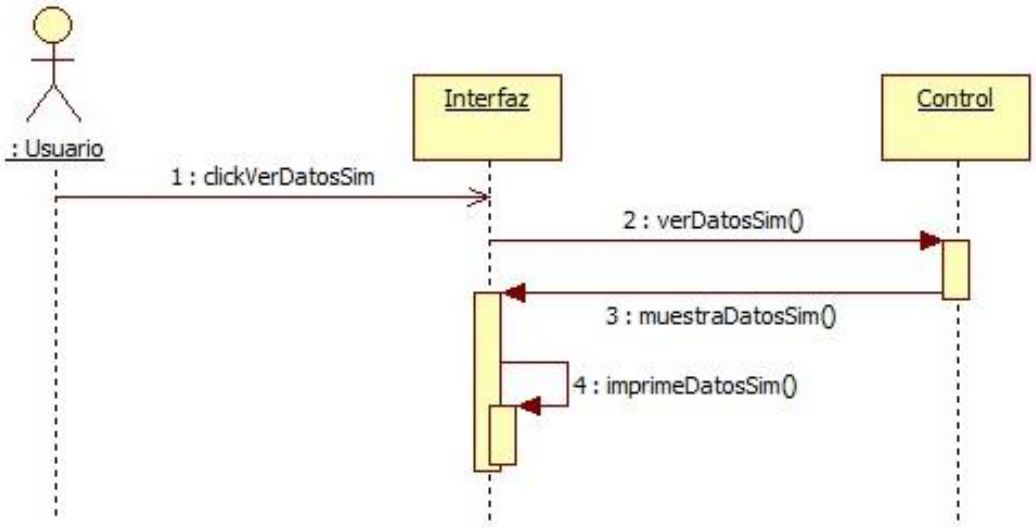


Ilustración 7. Diagrama de secuencia "Visualizar datos simulación"

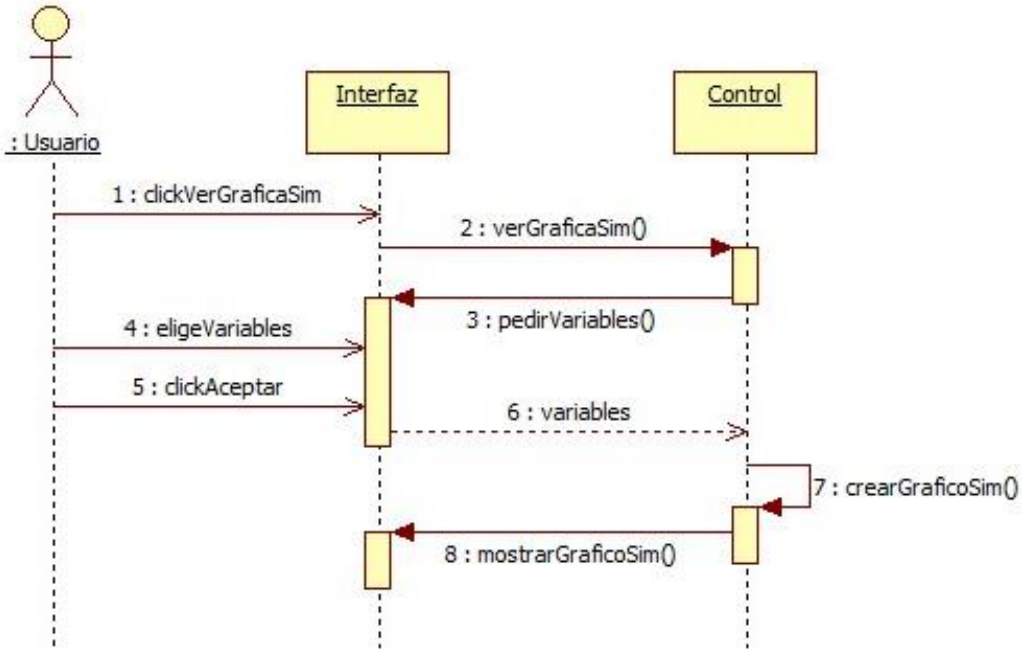


Ilustración 8. Diagrama de secuencia "Visualizar gráfica simulación"

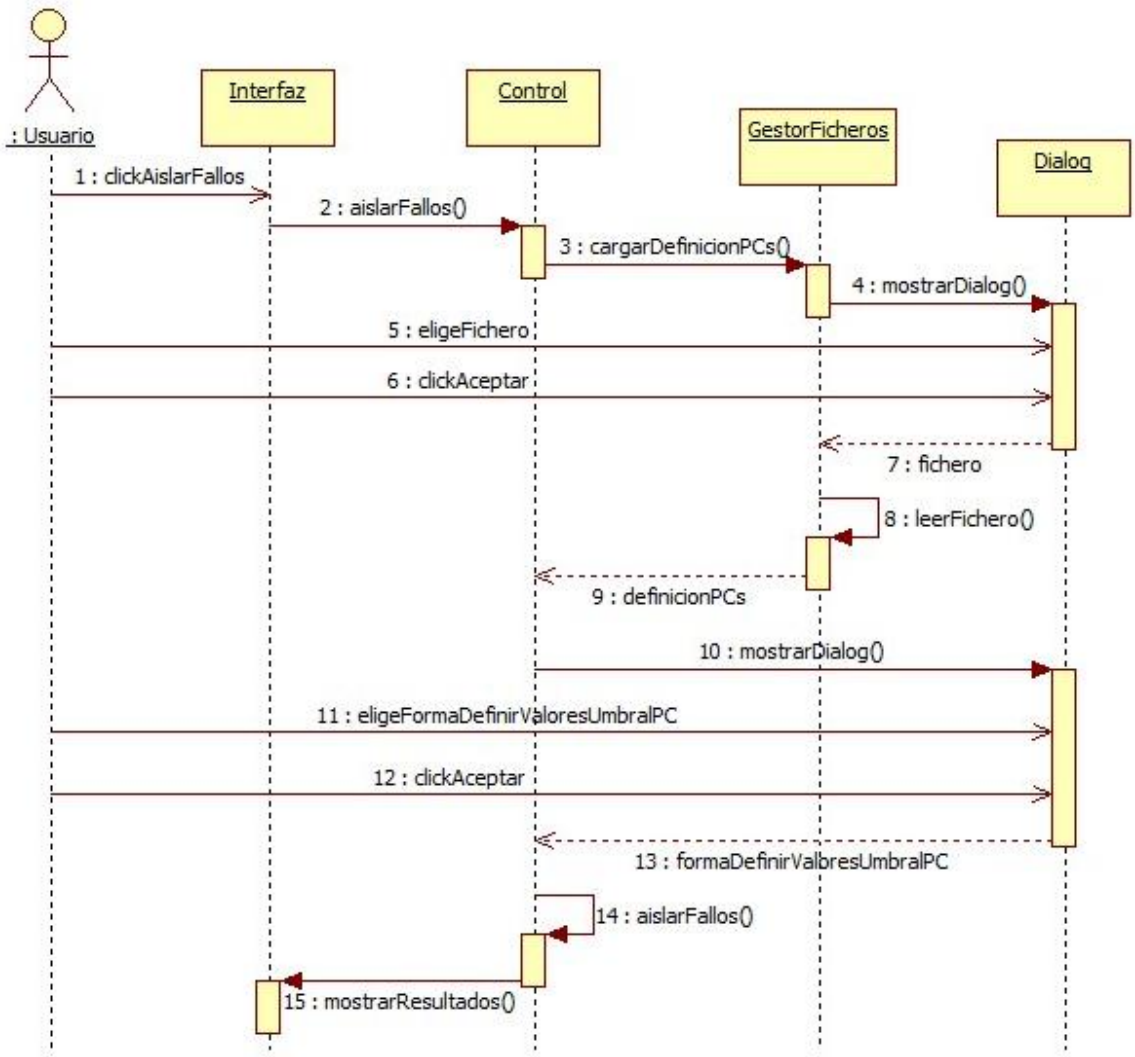


Ilustración 9. Diagrama de secuencia "Cargar algoritmo aislamiento de fallos"

Capítulo 5
Diseño

En este capítulo se presenta el diagrama de clases de diseño, la arquitectura y el diseño de la interfaz de la aplicación.

5.1. Diagrama de clases

El diagrama de clases de diseño que se muestra en la Ilustración 10 ha sido elaborado para tener en cuenta los detalles concretos de la implementación del sistema ha desarrollar. Este diagrama contiene las mismas clases que el diagrama elaborado en la fase de Análisis con la diferencia que el diagrama de clases de diseño añade a cada una de estas clases sus atributos y sus métodos.

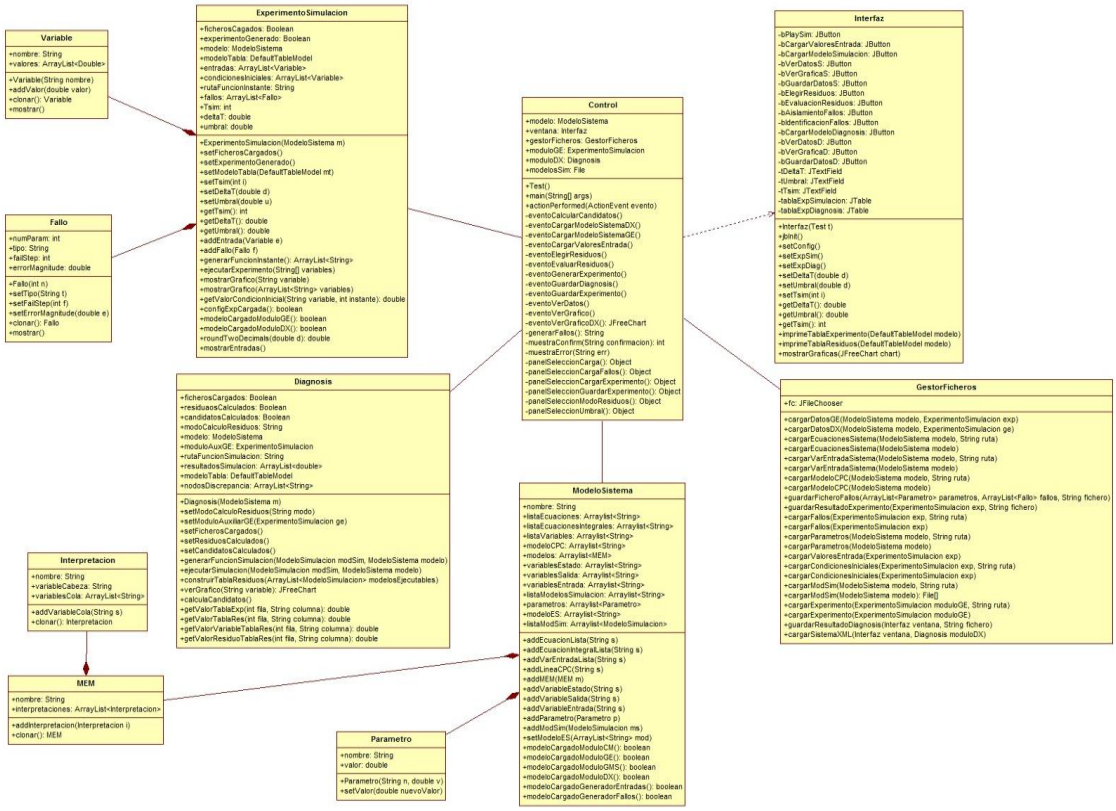


Ilustración 10. Diagrama de clases de diseño

5.2. Arquitectura

La arquitectura lógica es la que muestra los componentes lógicos que aparecen en la aplicación y la relación entre ellos. En este caso la arquitectura lógica es la correspondiente a la Ilustración 11.

La base de esta arquitectura es la máquina virtual de Java (JVM), ya que debe estar instalada en el ordenador para que la aplicación funcione. Sobre la base se encuentra el modelo que está compuesto por Simulación, Diagnosis y Modelo Sistema. El Modelo Sistema es el modelo de un experimento que contiene todos los algoritmos y parámetros necesarios. Simulación es el modelo de un experimento de simulación y Diagnosis es el modelo de un experimento de diagnosis.

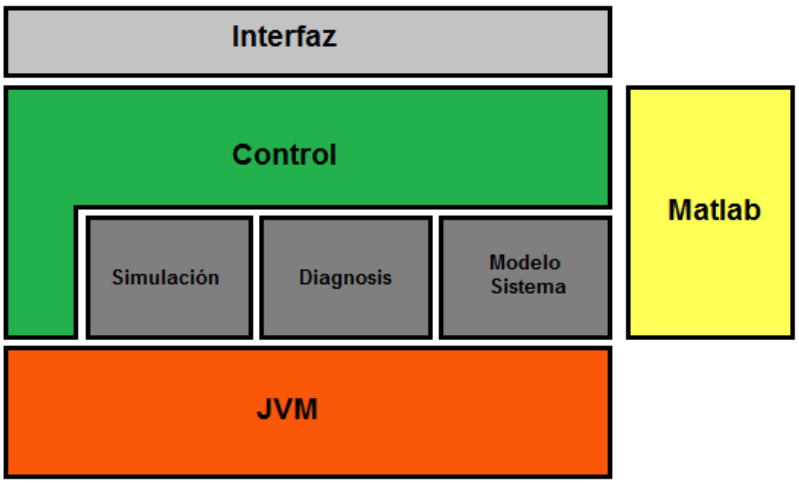


Ilustración 11. Arquitectura software

Por encima y envolviendo el modelo se encuentra Control que es el que se encarga de controlar todo lo que ocurre en la aplicación, tanto conectar con el modelo, como conectar con Matlab o como mostrar la interfaz. Se encarga de conectar internamente con Matlab para poder realizar los experimentos de simulación y diagnosis, gestionando el intercambio de información, de manera que pueda capturar instantáneamente los resultados de los experimentos.

Sobre todo esto se encuentra la Interfaz, que recupera los datos necesarios y los muestra de una manera adecuada.

5.3. Diseño interfaz

En este apartado se va a explicar el contenido de la interfaz. Como se muestra en la Ilustración 12 el diseño de la interfaz es muy sencillo.

Nombre	*Interfaz*
Descripción	Esta ventana es la principal en la que se engloba todo. En la parte izquierda superior se encuentra la zona de configuración, en la que se podrán modificar los valores por defecto de configuración de experimentos. A su lado se encuentra el botón para comenzar la simulación de modelos. En la parte central de la ventana se encuentran los botones con los que el usuario podrá realizar los experimentos. Existen dos grupos de botones, los que corresponden a la simulación y los que corresponden a la diagnosis. En la parte inferior de la ventana se encuentran unas pestañas donde el usuario podrá ver los resultados de los experimentos de simulación y diagnosis. En la Ilustración 12 se puede ver la pestaña de simulación, en la que se mostrarán los resultados de los experimentos de simulación en forma de tabla.
Activación	Esta ventana se activa al abrir la aplicación.

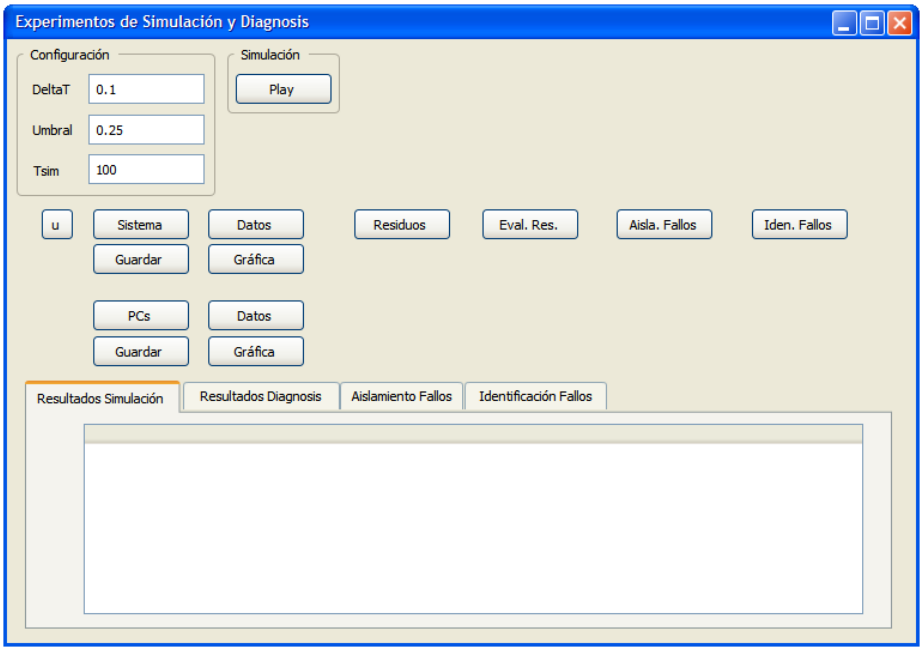
Boceto	
Eventos	Los eventos lanzados son: - eventoCargarValoresEntrada() - eventoCargarModeloSistemaGE() - eventoGenerarExperimento() - eventoVerDatos() - eventoVerGrafico() - eventoGuardarExperimento() - eventoCargarModeloSistemaDX() - eventoElegirResiduos() - eventoEvaluarResiduos() - eventoVerTablaResiduos() - eventoVerGraficoDX() - eventoCalcularCandidatos() - eventoGuardarDiagnosis()

Ilustración 12. Interfaz

Tabla 58. Interfaz

Nombre	*Pestaña de resultados de diagnosis*
Descripción	Esta interfaz se corresponde con la pestaña de diagnosis, en la que se mostrarán los resultados de los experimentos de diagnosis en forma de tabla y de gráfica.
Activación	Esta interfaz se activa cuando se pulsa encima del nombre de la pestaña.

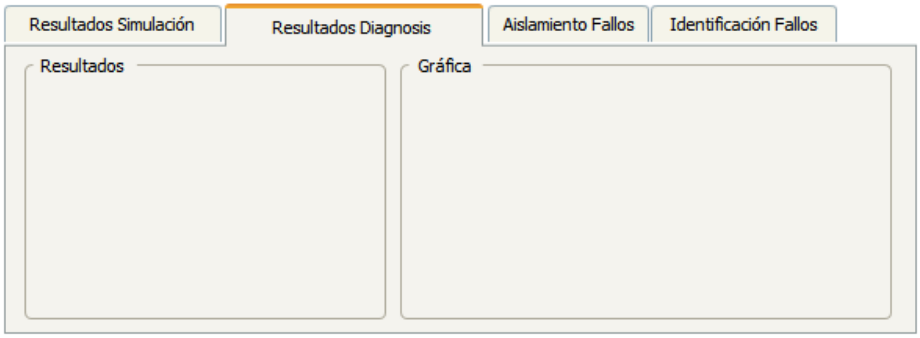
Boceto	
Eventos	Los eventos lanzados son: - imprimeTablaResiduos() - mostrarVerGraficoDX()

Ilustración 13. Pestaña de resultados diagnosis

Tabla 59. Pestaña de resultados de diagnosis

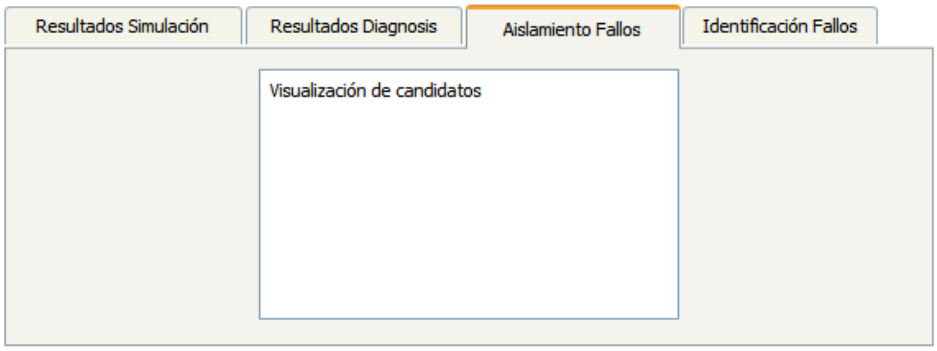
Nombre	*Pestaña de resultados de aislamiento de fallos*
Descripción	Esta interfaz pertenece a la pestaña de aislamiento, en la que se mostrarán los resultados del aislamiento de fallos en forma de texto.
Activación	Esta interfaz se activa cuando se pulsa encima del nombre de la pestaña.
Boceto	
Eventos	El evento lanzado es: - mostrarCandidatos()

Ilustración 14. Pestaña de resultados de aislamiento de fallos

Tabla 60. Pestaña de resultados de aislamiento de fallos

Capítulo 6
Pruebas

Este capítulo engloba las pruebas que han sido realizadas durante y después del desarrollo con el objetivo de detectar los posibles errores. Las pruebas realizadas han sido unitarias, de integración y de sistema.

Las pruebas unitarias son una forma de comprobar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. El objetivo de estas pruebas es aislar cada parte del programa y mostrar que las partes individuales son correctas. Estas pruebas no descubren todos los errores de código y sólo son efectivas si se usan en conjunto con otras pruebas de software.

Las pruebas de integración se realizan una vez finalizadas las pruebas unitarias. Las pruebas de integración combinan los módulos de código y los prueba como un grupo. Estas pruebas verifican el correcto funcionamiento de los componentes de la aplicación actuando en conjunto.

Las pruebas de sistema son aquellas cuyo objetivo es probar todo el sistema software completo e integrado, normalmente desde el punto de vista de los requisitos.

La realización de las pruebas unitarias, de integración y de sistema se han llevado a cabo desde dos perspectivas diferentes:

- Las **pruebas de caja blanca** son aquellas que se realizan sobre funciones internas de un módulo. Se comprueba así la aplicación a nivel de código fuente.

- Las **pruebas de caja negra** son aquellas que se llevan a cabo sobre la interfaz de la aplicación, obviando el comportamiento interno y la estructura del programa.

6.1.Pruebas de caja blanca

Las pruebas de caja blanca se han ido realizando a la vez que el desarrollo del código, de forma que cada método y cada clase han sido probadas unitariamente. Además, han sido probados los flujos entre métodos y clases durante la integración.

A medida que se probaba el código se ha intentado recorrer todos los posibles caminos de ejecución. Por ejemplo, revisando el uso de las variables o comprobando los bucles.

6.2.Pruebas de caja negra

Las pruebas de caja negra se centran en el estudio de las entradas y salidas de un módulo sin tener en cuenta su funcionamiento interno, comprobando que los requisitos funcionales se han cumplido. No son una alternativa a las pruebas de caja blanca sino que se han realizado las dos por separado y de forma complementaria, para detectar diferentes tipos de errores. Se han realizado pruebas de integración y de sistema con este modelo de pruebas de caja negra.

De todos los casos de prueba (CP) de caja negra que se han realizado, se van a mostrar algunos de ellos.

CP01	Cargar entradas
Propósito	Comprobar que se ha cargado el fichero con las entradas.
Datos de entrada	entrada.txt
Acción esperada	Botón "Cargar Modelo Simulación" habilitado.
Resultado	Correcto.

Tabla 61. CP01 - Cargar entradas

CP02	Cargar modelo simulación
Propósito	Comprobar que se han cargado los ficheros con el modelo del sistema, las variables de entrada, los parámetros, las condiciones iniciales y los fallos paramétricos.
Datos de entrada	EDOs.m, variables.txt, parametros.txt, CondicionesIniciales.txt, fallos.txt
Acción esperada	Botón "Play" de simulación habilitado.
Resultado	Correcto.

Tabla 62. CP02 - Cargar modelo simulación

CP03	Ejecutar simulación
Propósito	Comprobar que se ha ejecutado correctamente el experimento de simulación.
Datos de entrada	Resultados del experimento de simulación.
Acción esperada	Botones "Ver Datos", "Ver Gráfica", "Guardar Datos Simulación" y "Cargar Modelo Diagnosis" habilitados.
Resultado	Correcto.

Tabla 63. CP03 - Ejecutar simulación

CP04	Ver datos simulación
Propósito	Comprobar que se muestran los resultados del experimento de simulación en forma de tabla.
Datos de entrada	Resultados del experimento de simulación.
Acción esperada	Muestra una tabla con los resultados del experimento de simulación.
Resultado	Correcto.

Tabla 64. CP04 - Ver datos simulación

CP05	Ver gráfica simulación
Propósito	Comprobar que se muestran los resultados del experimento de simulación en forma de gráfica.
Datos de entrada	Resultados del experimento de simulación y variables.
Acción esperada	Muestra una gráfica con los resultados del experimento de simulación.
Resultado	Correcto.

Tabla 65. CP05 - Ver gráfica simulación

CP06	Guardar datos simulación
Propósito	Comprobar que se guardan los resultados del experimento de simulación en un fichero .txt.
Datos de entrada	Resultados del experimento de simulación.
Acción esperada	Crea un fichero con los resultados del experimento de simulación.
Resultado	Correcto.

Tabla 66. CP06- Guardar datos simulación

CP07	Cargar modelo diagnosis
Propósito	Comprobar que se han cargado los ficheros con las variables de entrada, los modelos de simulación, los parámetros, las condiciones iniciales, el resultado del cálculo de PCs y el resultado del experimento de simulación.
Datos de entrada	variables.txt, parametros.txt, Modelo1(CEM1)Simulacion.sim, Modelo2(CEM10)Simulacion.sim, Modelo3(CEM13)Simulacion.sim, CondicionesIniciales.txt, ModeloCPC.txt, resultadoSimulacion.txt
Acción esperada	Botón "Elegir Residuos" habilitado.
Resultado	Correcto.

Tabla 67. CP07 - Cargar modelo diagnosis

CP08	Elegir residuos
Propósito	Comprobar que se ha cargado el algoritmo de evaluación de residuos.
Datos de entrada	Algoritmo de evaluación de residuos.
Acción esperada	Botón "Evaluar Residuos" habilitado.
Resultado	Correcto.

Tabla 68. CP08 - Elegir residuos

CP09	Evaluar residuos
Propósito	Comprobar que se han evaluado los de residuos.
Datos de entrada	Algoritmo de evaluación de residuos.
Acción esperada	Botones "Ver Datos", "Ver Gráfica" y "Aislamiento de Fallos" habilitados.
Resultado	Correcto.

Tabla 69. CP09 - Evaluar residuos

CP10	Ver datos diagnosis
Propósito	Comprobar que se muestran los resultados del experimento de diagnosis en forma de tabla.
Datos de entrada	Resultados del experimento de diagnosis.
Acción esperada	Muestra una tabla con los resultados del experimento de diagnosis.
Resultado	Correcto.

Tabla 70. CP10 - Ver datos diagnosis

CP11	Ver gráfica diagnosis
Propósito	Comprobar que se muestran los resultados del experimento de diagnosis en forma de gráfica.
Datos de entrada	Resultados del experimento de diagnosis.
Acción esperada	Muestra una gráfica con los resultados del experimento de diagnosis.
Resultado	Correcto.

Tabla 71. CP11 - Ver gráfica diagnosis

CP12	Aislar fallos
Propósito	Comprobar que se muestran los resultados del aislamiento de fallos.
Datos de entrada	Resultados del experimento de diagnosis y sistema_3_tanques.xml.
Acción esperada	Muestra los resultados del aislamiento de fallos y el botón "Guardar Datos Diagnosis" es habilitado.
Resultado	Correcto.

Tabla 72. CP12 - Aislar fallos

CP13	Guardar datos diagnosis
Propósito	Comprobar que se guardan los resultados del experimento de diagnosis en un fichero .txt.
Datos de entrada	Resultados del experimento de diagnosis.
Acción esperada	Crea un fichero con los resultados del experimento de diagnosis.
Resultado	Correcto.

Tabla 73. CP13 - Guardar datos diagnosis

Capítulo 7
Manuales

7.1.Manual de instalación

Para el correcto funcionamiento de la aplicación es necesario copiar la carpeta Aplicación en el disco del ordenador y tener instalado:

- La herramienta Matlab en su versión 8.2 (R2013b) o superior.
- El software JDK 1.7 o superior. Este puede obtenerse en la página oficial de Oracle:

  http://www.oracle.com/technetwork/es/java/javase/downloads/jdk7-downloads-1880260.html

Además, la aplicación hace uso de archivos de rutas, que son ficheros de texto con extensión *.txt* que contienen las rutas absolutas de los ficheros de entrada necesarios

para llevar a cabo los experimentos de simulación y diagnosis. En este fichero se tienen que actualizar las rutas de los archivos si éstas fueran modificadas. Un ejemplo de archivo de rutas es el que aparece en la Ilustración 15. Los ficheros de entrada de los que se especifica su ruta son:

- EDOs: es un fichero con extensión *.m* (ejecutable por Matlab) que contiene las ecuaciones del sistema.

- Variables: es un fichero con extensión *.txt* que contiene las variables del sistema divididas en cuatro grupos: de estado, de salida, de entrada y parámetros.

- Condiciones iniciales: es un fichero con extensión *.txt* que contiene los valores de cada variable y parámetro del sistema en instantes anteriores a cero.

- Modelo de Cálculo de PCs (CPCs): es un fichero con extensión *.txt* que contiene el modelo de los resultados del cálculo de Posibles Conflictos.

- Datos de entada: es un fichero con extensión *.txt* que contiene los valores de las variables de entrada para cada instante de simulación.

- Parámetros: es un fichero con extensión *.txt* que contiene los valores de los parámetros de sistema.

- Fallos: es un fichero con extensión *.txt* que contiene una descripción de los fallos que se van a producir durante la ejecución del experimento. Para cada fallo se indica el identificador del parámetro, el tipo de fallo (ADD o MULT), el instante en que se produce y su magnitud.

- Resultados del experimento: es un fichero con extensión *.txt* que contiene en forma de tabla los resultados de la ejecución de un experimento de simulación.

- Modelos de simulación: son ficheros con extensión *.sim* que contienen los modelos de simulación de cada PC existente.

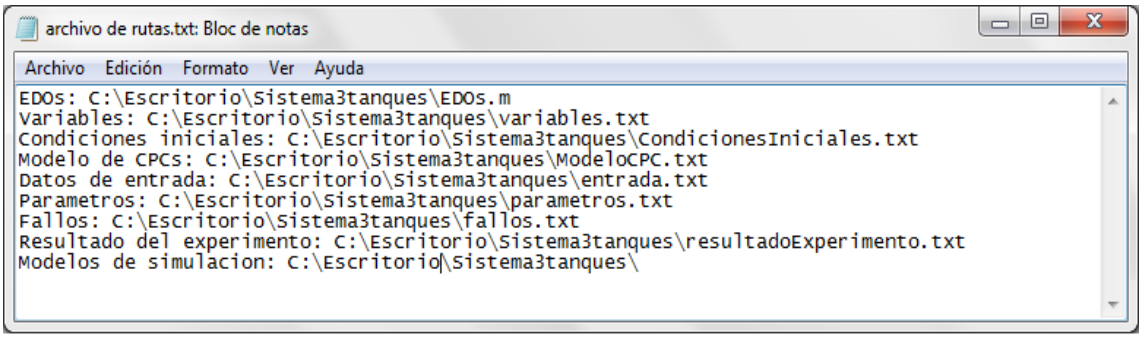


Ilustración 15. Ejemplo de archivo de rutas

Una vez que tenemos todo el software indispensable instalado y todos los ficheros necesarios, ya podemos abrir nuestra aplicación que no necesita instalación, simplemente es hacer doble click sobre el archivo ejecutable.

7.2.Manual de usuario

Esta aplicación tiene una interfaz sencilla e intuitiva como se puede ver en la Ilustración 16. Para hacer un correcto uso de ésta y poder realizar la ejecución de un experimento de simulación y un experimento de diagnosis adecuadamente, habría que seguir los siguientes pasos:

- Paso 1. Configurar el experimento.
- Paso 2. Cargar y ejecutar el experimento de simulación.
    - Paso 2.1. Cargar el fichero con los datos de entrada.
    - Paso 2.2. Cargar los ficheros para el experimento de simulación.
    - Paso 2.3. Ejecutar el experimento de simulación.
- Paso 3. Visualización de resultados del experimento de simulación.
    - Paso 3.1. Ver los resultados del experimento de simulación.
    - Paso 3.2. Ver gráficamente los resultados del experimento de simulación.
- Paso 4. Guardar los resultados del experimento de simulación.
- Paso 5. Cargar y ejecutar el experimento de diagnosis.
    - Paso 5.1. Cargar los ficheros para el experimento de diagnosis.
    - Paso 5.2. Elegir la forma de calcular residuos.
    - Paso 5.3. Evaluar residuos.
- Paso 6. Visualización de resultados del experimento de diagnosis.
    - Paso 6.1.Ver los resultados del experimento de diagnosis.
    - Paso 6.2. Ver gráficamente los resultados del experimento de diagnosis.
- Paso 7. Aislar fallos.
- Paso 8. Guardar los resultados del experimento de diagnosis.

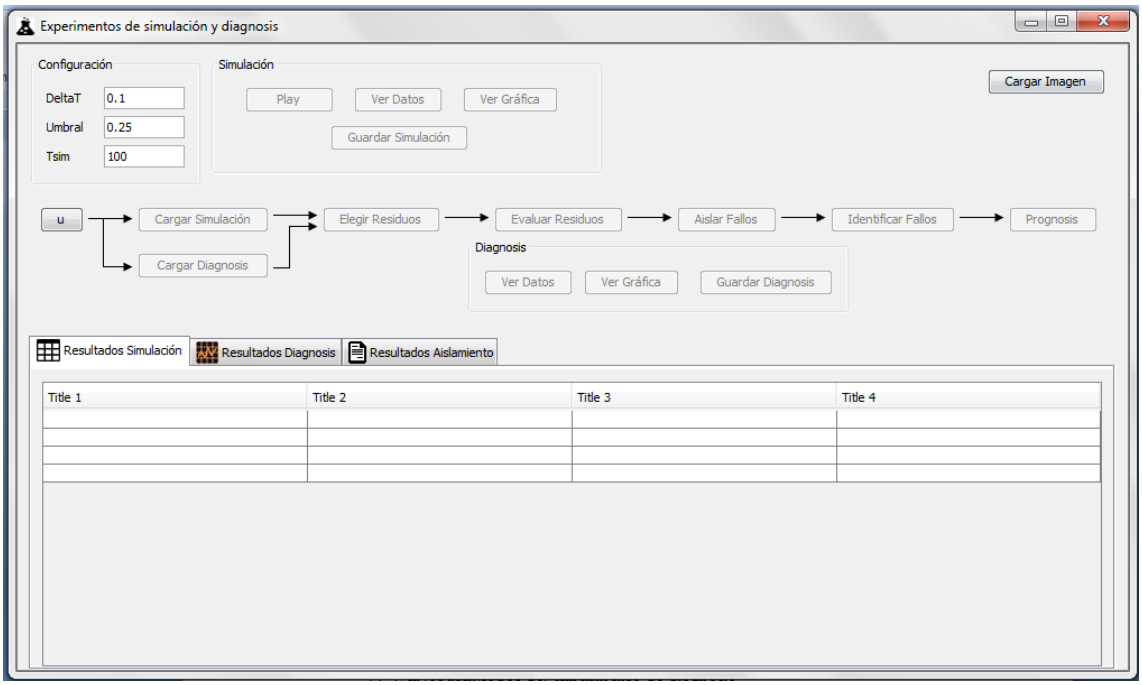


Ilustración 16. Ventana principal

Paso 1. Configurar el experimento.

Si se desea modificar los valores establecidos por defecto en la configuración del experimento, únicamente deberá cambiar los valores deseados que aparecen en la parte reservada para la configuración (Ilustración 17).

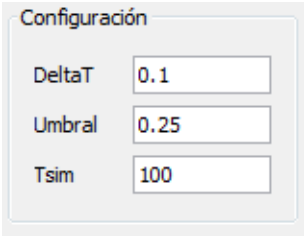


Ilustración 17. Configuración

Paso 2. Cargar y ejecutar el experimento de simulación.

Una vez establecidos los datos de configuración, se procede a cargar y ejecutar el experimento de simulación.

Paso 2.1. Cargar el fichero con los datos de entrada.

Para cargar el fichero con los datos de entrada se debe pulsar el botón "u". Se abre una ventana con el explorador de archivos en la que debemos buscar y seleccionar el fichero con extensión *.txt* que contiene los valores de las variables de entrada para cada instante de simulación, como se muestra en la Ilustración 18.

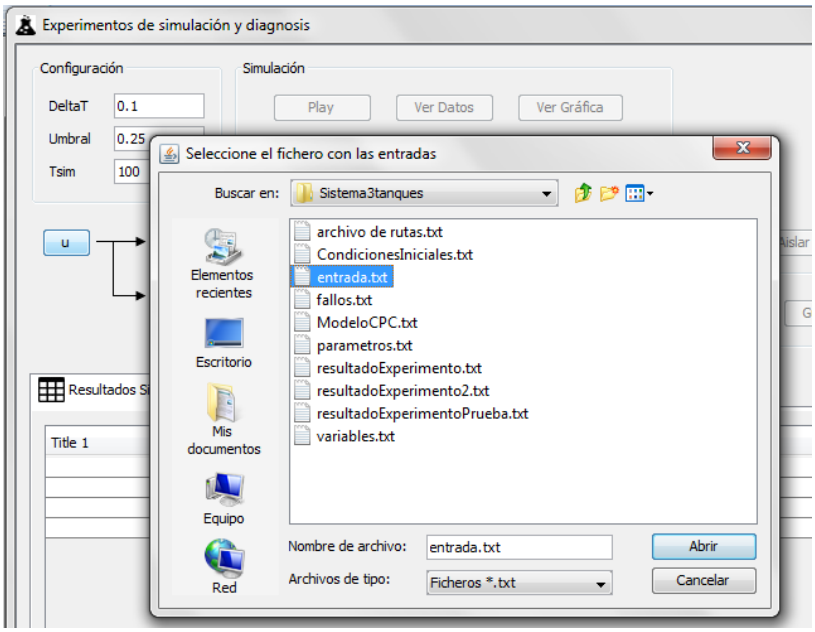


Ilustración 18. Cargar fichero entradas

Paso 2.2. Cargar los ficheros para el experimento de simulación.

Una vez cargado el fichero con los datos de entrada, automáticamente se habilita el botón "Cargar Simulación", que al pulsarle nos muestra una nueva ventana en la que hay que elegir la forma en la que se quieren cargar los datos (Ilustración 19), ya que existen dos formas de cargar los ficheros necesarios para la simulación.

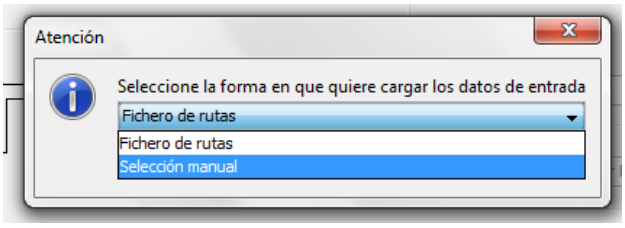


Ilustración 19. Elegir forma de cargar ficheros

Si elegimos "Fichero de rutas", se abre una ventana con el explorador de archivos en la que debemos buscar y seleccionar el fichero con las rutas de los archivos necesarios para la simulación.

Si elegimos "Selección manual", se abre una nueva ventana para cargar cada uno de los archivos necesarios para la simulación. Para cargar el modelo del sistema (EDOs), las variables de entrada, los parámetros y las condiciones iniciales se hace mediante una ventana con el explorador de archivos en la que debemos buscar y seleccionar el fichero deseado de una forma similar a como se carga el fichero con los datos de entrada. Para cargar los fallos paramétricos, aparece una nueva ventana en la que hay que elegir la forma en la que se quieren cargar los fallos (Ilustración 20), ya que existen dos formas de cargar los fallos paramétricos.

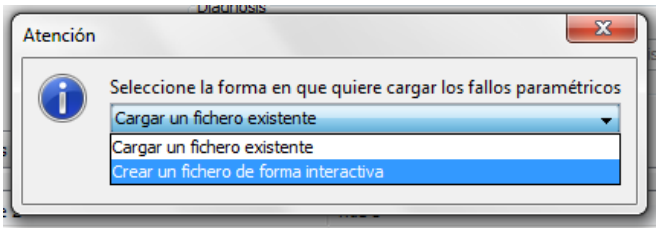


Ilustración 20. Elegir forma de cargar fichero de fallos

Si seleccionamos "Cargar un fichero existente", se abre una ventana con el explorador de archivos en la que debemos buscar y seleccionar el fichero con los fallos paramétricos, como sucede con el resto de ficheros.

Si seleccionamos "Crear un fichero de forma interactiva", se abre una nueva ventana (Ilustración 21) en la que debemos elegir los parámetros del experimento que van a causar fallo.

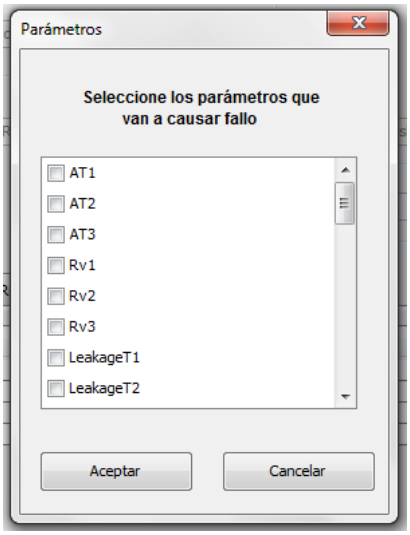


Ilustración 21. Elegir parámetros

Una vez elegidos los parámetros que van a causar fallo, aparece una ventana (Ilustración 22) para cada fallo, en la que debemos insertar el instante en que va a suceder el fallo, su magnitud y el tipo de fallo.

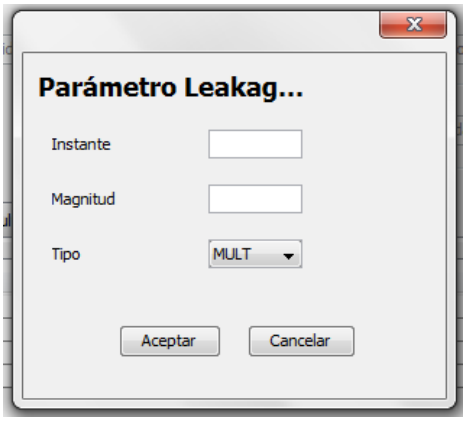


Ilustración 22. Datos fallo

Seguidamente aparece una ventana con el explorador de archivos en la que debemos elegir la carpeta en que queramos guardar el archivo y darlo un nombre.

Paso 2.3. Ejecutar el experimento de simulación.

Después de cargar todos los ficheros necesarios para la simulación, se habilita el botón "Play" que debemos pulsar para que comience la ejecución del experimento de simulación. Antes de comenzar la simulación los botones de "Ver Datos", "Ver Gráfica" y "Guardar Simulación" estarán deshabilitados y una vez que terminada se habilitarán.

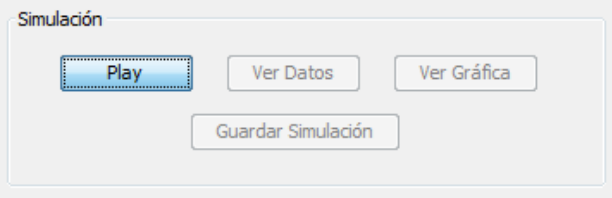


Ilustración 23. Botones de simulación deshabilitados

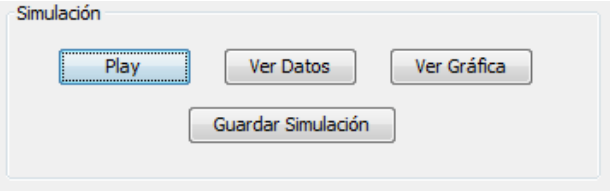


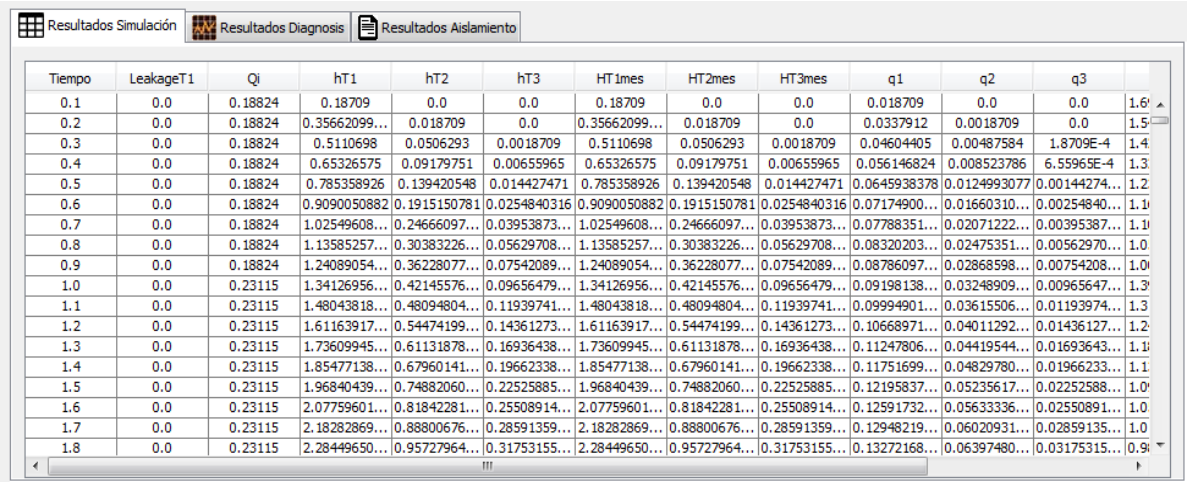
Ilustración 24. Botones de simulación habilitados

Paso 3. Visualización de resultados del experimento de simulación.

Posteriormente, se puede visualizar los resultados del experimento de simulación en forma de tabla y/o de gráfica.

Paso 3.1. Ver los resultados del experimento de simulación.

Para visualizar los resultados del experimento de simulación en forma de tabla, se debe pulsar el botón "Ver Datos" (Ilustración 24). Estos datos serán mostrados en la pestaña de "Resultados Simulación" como se muestra en Ilustración 25.



Tiempo	LeakageT1	Qi	hT1	hT2	hT3	HT1mes	HT2mes	HT3mes	q1	q2	q3	
0.1	0.0	0.18824	0.18709	0.0	0.0	0.18709	0.0	0.0	0.018709	0.0	0.0	1.6
0.2	0.0	0.18824	0.35662099...	0.018709	0.0	0.35662099...	0.018709	0.0	0.0337912	0.0018709	0.0	1.5
0.3	0.0	0.18824	0.5110698	0.0506293	0.0018709	0.5110698	0.0506293	0.0018709	0.04604405	0.00487584	1.8709E-4	1.4
0.4	0.0	0.18824	0.65326575	0.09179751	0.00655965	0.65326575	0.09179751	0.00655965	0.056146824	0.008523786	6.55965E-4	1.3
0.5	0.0	0.18824	0.785358926	0.139420548	0.014427471	0.785358926	0.139420548	0.014427471	0.0645938378	0.0124993077	0.00144274...	1.2
0.6	0.0	0.18824	0.9090050882	0.1915150781	0.0254840316	0.9090050882	0.1915150781	0.0254840316	0.07174900...	0.01660310...	0.00254840...	1.1
0.7	0.0	0.18824	1.02549608...	0.24666097...	0.03953873...	1.02549608...	0.24666097...	0.03953873...	0.07788351...	0.02071222...	0.00395387...	1.1
0.8	0.0	0.18824	1.13585257...	0.30383226...	0.05629708...	1.13585257...	0.30383226...	0.05629708...	0.08320203...	0.02475351...	0.00562970...	1.0
0.9	0.0	0.18824	1.24089054...	0.36228077...	0.07542089...	1.24089054...	0.36228077...	0.07542089...	0.08786097...	0.02868598...	0.00754208...	1.0
1.0	0.0	0.23115	1.34126956...	0.42145576...	0.09656479...	1.34126956...	0.42145576...	0.09656479...	0.09198138...	0.03248909...	0.00965647...	1.3
1.1	0.0	0.23115	1.48043818...	0.48094804...	0.11939741...	1.48043818...	0.48094804...	0.11939741...	0.09994901...	0.03615506...	0.01193974...	1.3
1.2	0.0	0.23115	1.61163917...	0.54474199...	0.14361273...	1.61163917...	0.54474199...	0.14361273...	0.10668971...	0.04011292...	0.01436127...	1.2
1.3	0.0	0.23115	1.73609945...	0.61131878...	0.16936438...	1.73609945...	0.61131878...	0.16936438...	0.11247806...	0.04419544...	0.01693643...	1.1
1.4	0.0	0.23115	1.85477138...	0.67960141...	0.19662338...	1.85477138...	0.67960141...	0.19662338...	0.11751699...	0.04829780...	0.01966233...	1.1
1.5	0.0	0.23115	1.96840439...	0.74882060...	0.22525885...	1.96840439...	0.74882060...	0.22525885...	0.12195837...	0.05235617...	0.02252588...	1.0
1.6	0.0	0.23115	2.07759601...	0.81842281...	0.25508914...	2.07759601...	0.81842281...	0.25508914...	0.12591732...	0.05633336...	0.02550891...	1.0
1.7	0.0	0.23115	2.18282869...	0.88800676...	0.28591359...	2.18282869...	0.88800676...	0.28591359...	0.12948219...	0.06020931...	0.02859135...	1.0
1.8	0.0	0.23115	2.28449650...	0.95727964...	0.31753155...	2.28449650...	0.95727964...	0.31753155...	0.13272168...	0.06397480...	0.03175315...	0.9

Ilustración 25. Resultados experimento de simulación

Paso 3.2. Ver gráficamente los resultados del experimento de simulación.

Los resultados del experimento de simulación se pueden visualizar en forma de gráfica. Para ello, se debe pulsar el botón "Ver Gráfica" (Ilustración 24) y se muestra una ventana (Ilustración 26) en la que debemos seleccionar las variables que deseemos que se presenten en la gráfica que aparecerá en una ventana nueva (Ilustración 27).

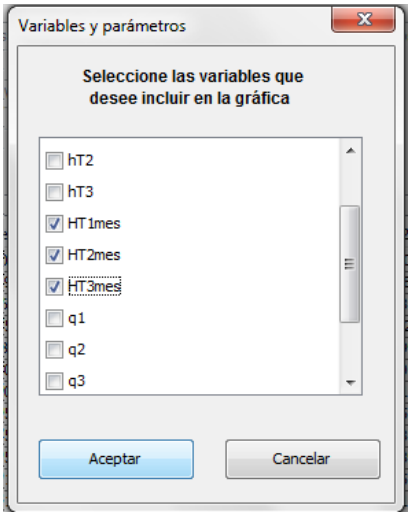


Ilustración 26. Selección de variables

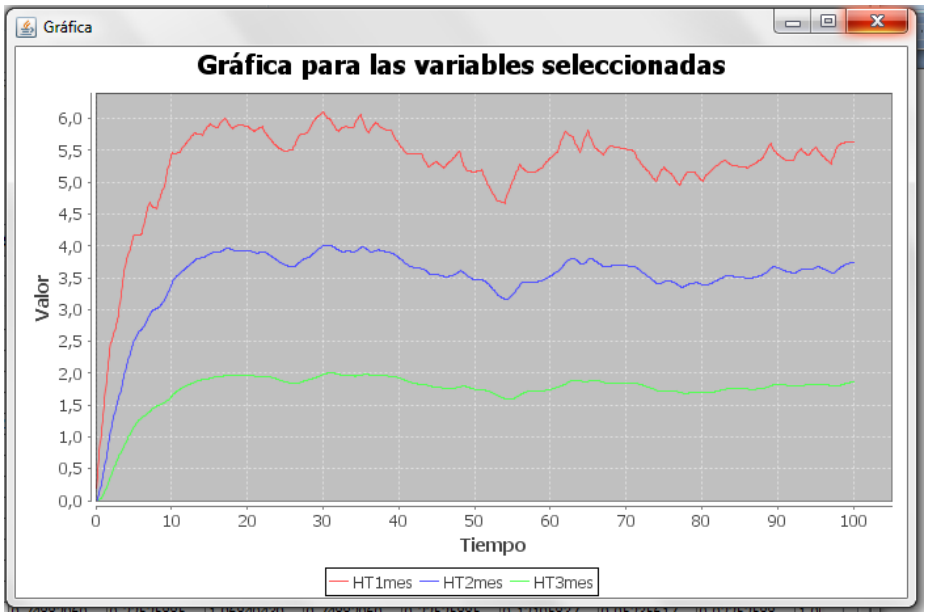


Ilustración 27. Gráfica resultados simulación

Paso 4. Guardar los resultados del experimento de simulación.

Como último paso del experimento de simulación, debemos guardar los resultados obtenidos. Para ello, debemos pulsar el botón "Guardar Simulación" y aparece una ventana con el explorador de archivos en la que debemos elegir la carpeta en que queramos guardar el archivo y darlo un nombre.

Paso 5. Cargar y ejecutar el experimento de diagnosis.

Al finalizar la ejecución del experimento de simulación se habilitó el botón "Cargar Diagnosis". Una vez guardados los resultados del experimento de simulación, se procede a cargar y ejecutar el experimento de diagnosis.

Paso 5.1. Cargar los ficheros para el experimento de diagnosis.

Pulsando el botón "Cargar Diagnosis" se muestra una nueva ventana en la que hay que elegir la forma en la que se quieren cargar los datos (Ilustración 19), ya que existen dos formas de cargar los ficheros necesarios para la simulación.

Si elegimos "Fichero de rutas", se abre una ventana con el explorador de archivos en la que debemos buscar y seleccionar el fichero con las rutas de los archivos necesarios para la diagnosis.

Si elegimos "Selección manual", se abre una nueva ventana para cargar cada uno de los archivos necesarios para la diagnosis. Para cargar las variables de entrada, los modelos de simulación (ficheros con extensión .sim), los parámetros, las condiciones iniciales, el resultado del cálculo de PCs y el resultado de un experimento de simulación, se hace mediante una ventana con el explorador de archivos en la que debemos buscar y seleccionar el fichero deseado de una forma similar a como se cargan los ficheros en la simulación.

Paso 5.2. Elegir la forma de calcular residuos.

Después de cargar los ficheros se habilita el botón "Elegir Residuos", que debemos pulsar para seleccionar la manera en que serán calculados los residuos, como se muestra en la Ilustración 28.

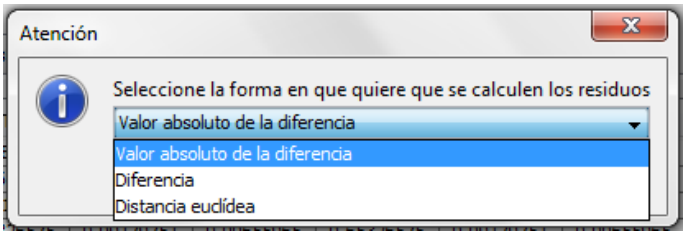


Ilustración 28. Elegir modo de calcular residuos

Paso 5.3. Evaluar residuos.

Se habilita el botón "Evaluar Residuos" una vez seleccionada la forma en la que se calcularán los residuos. Pulsando el botón se ejecuta el experimento de diagnosis y el cálculo de residuos. Además se habilitan los botones "Ver Datos", "Ver Gráfica" y "Aislar Fallos" (Ilustración 29).

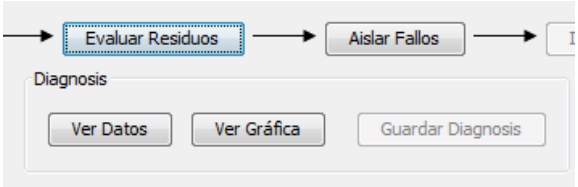


Ilustración 29. Botones diagnosis

Paso 6. Visualización de resultados del experimento de diagnosis.

Se puede visualizar los resultados del experimento de diagnosis en forma de tabla y/o de gráfica.

Paso 6.1.Ver los resultados del experimento de diagnosis.

Para visualizar los resultados del experimento de diagnosis en forma de tabla, se debe pulsar el botón "Ver Datos" (Ilustración 29). Estos datos se muestran en la pestaña de "Resultados Diagnosis" como se muestra en Ilustración 30.

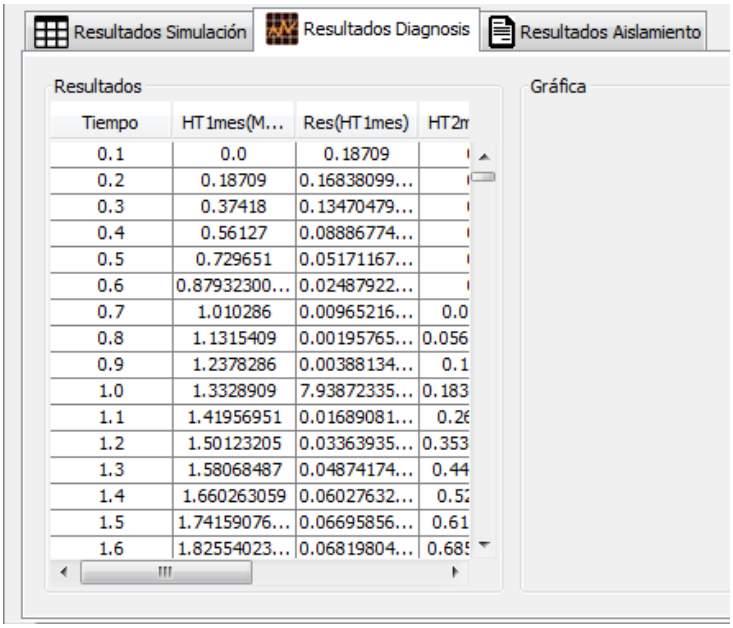


Ilustración 30. Tabla resultados diagnosis

Paso 6.2. Ver gráficamente los resultados del experimento de diagnosis.

Los resultados del experimento de diagnosis se pueden visualizar en forma de gráfica. Para ello, se debe pulsar el botón "Ver Gráfica" (Ilustración 29) y se muestra una ventana (Ilustración 31) en la que debemos seleccionar la variable que queremos evaluar en una gráfica, que se muestra en la pestaña de "Resultados Diagnosis" (Ilustración 32).

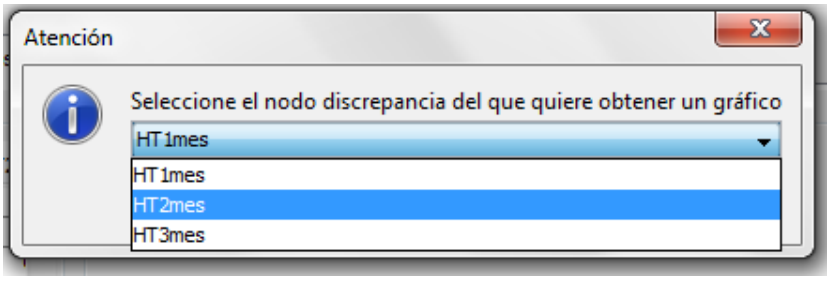


Ilustración 31. Selección nodos

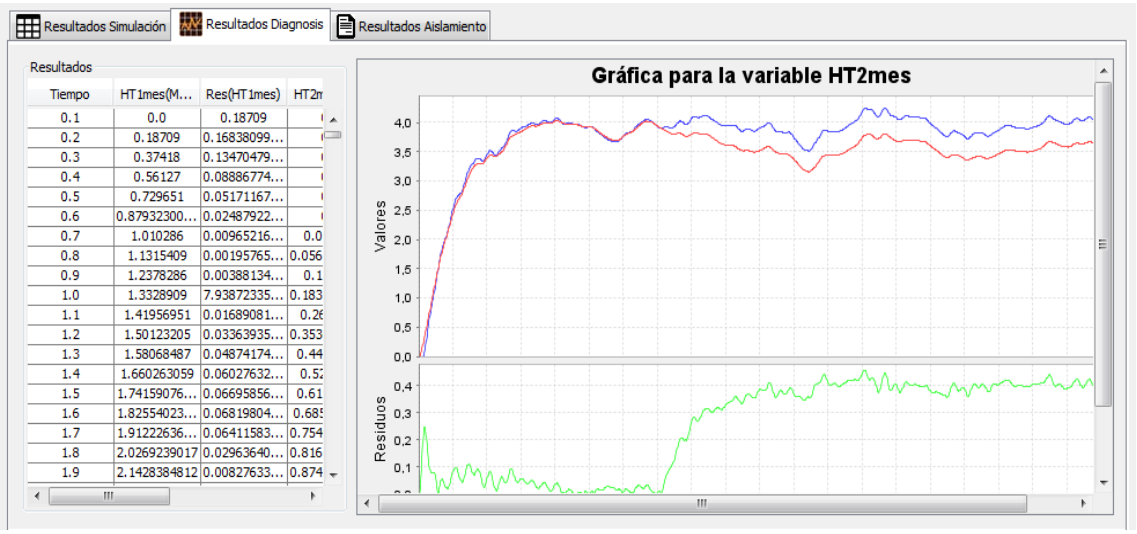


Ilustración 32. Gráfica diagnosis

Paso 7. Aislar fallos.

Una vez tenemos los residuos calculados pasamos a aislar los fallos. Para ello, debemos pulsar el botón "Aislar Fallos" (Ilustración 29) y la aplicación abre una ventana en la que hay que seleccionar el fichero .xml con la definición de PCs. Después nos aparece una pantalla en la que debemos seleccionar una de las tres formas de definir los valores umbral de cada PC, como se muestra en la Ilustración 33.

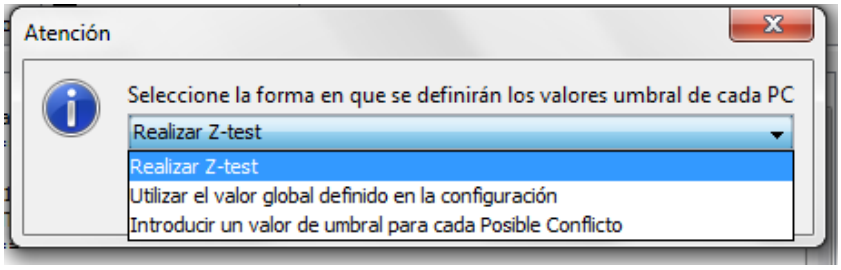


Ilustración 33. Selección valores umbral de cada PC

Al escoger la opción "Realizar Z-test", se muestra una ventana en la que se debe insertar los valores de los diferentes parámetros como se muestra en la Ilustración 34. Una vez insertados estos valores se muestra el resultado del aislamiento de fallos en la pestaña "Resultados Aislamiento" como se muestra en la Ilustración 35.

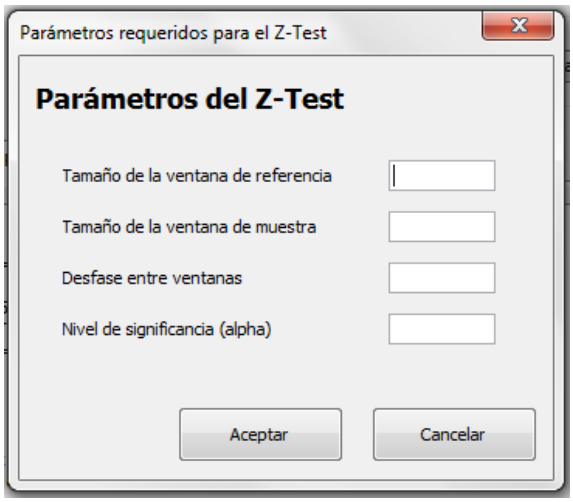


Ilustración 34. Parámetros del Z-test

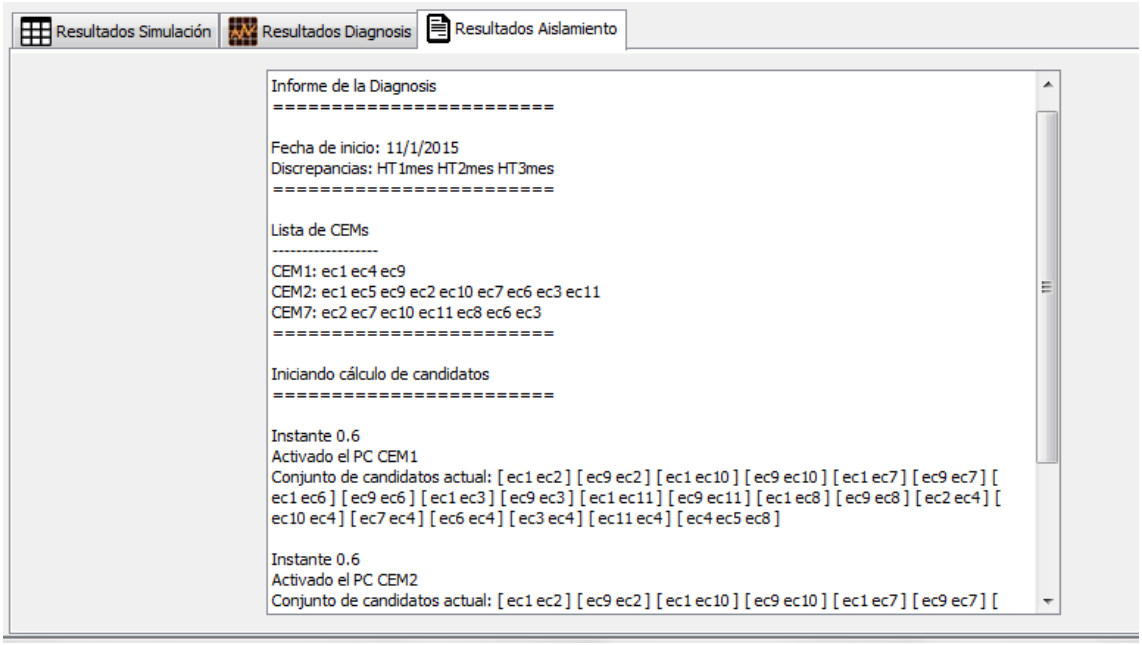


Ilustración 35. Resultados aislamiento de fallos

Si escogemos la opción "Utilizar el valor global definido en la configuración", se muestra el resultado del aislamiento de fallos en la pestaña "Resultados Aislamiento" como se muestra en la Ilustración 35.

Al escoger la opción "Introducir un valor de umbral para cada Posible Conflicto", se muestra una ventana en la que se debe insertar los valores umbral para cada elemento como se muestra en la Ilustración 36. Una vez insertados estos valores se muestra el resultado del aislamiento de fallos en la pestaña "Resultados Aislamiento" como se muestra en la Ilustración 35.

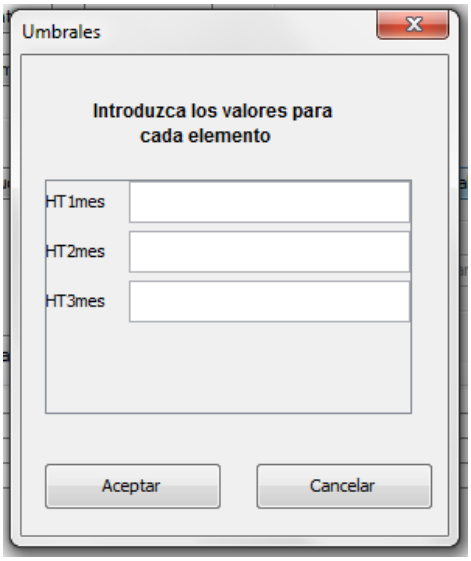


Ilustración 36. Valores umbral

Despues de realizar el aislamiento con cualquiera de las tres formas posibles se habilita el botón "Guardar Diagnosis".

Paso 8. Guardar los resultados del experimento de diagnosis.

Como último paso del experimento de diagnosis, debemos guardar los resultados obtenidos. Para ello, debemos pulsar el botón "Guardar Diagnosis" y aparece una ventana con el explorador de archivos en la que debemos elegir la carpeta en que queramos guardar el archivo y darlo un nombre.

Capítulo 8
Conclusiones

Por último, en este capítulo se van a mostrar las conclusiones generales después de realizar este TFG y el trabajo futuro que se podría hacer para ampliar la aplicación.

8.1. Conclusiones generales

Como ya se ha dicho a lo largo de este documento, con este Trabajo Fin de Grado se quería conseguir la implementación de un entorno integrado para la simulación de modelos y la realización de experimentos de diagnosis. Este objetivo se ha cumplido de manera satisfactoria, ya que hacía falta un gran conocimiento de Matlab para la implementación de una serie de algoritmos, una serie de modelos de simulación y una colección de técnicas de diagnosis de los que disponemos y que no están relacionados entre sí. Ahora con esta aplicación hemos conseguido unificarlos y no hace falta tener conocimientos de Matlab, aunque la aplicación la utilice.

La realización de este trabajo me ha permitido desarrollar una aplicación con bastante amplitud, trabajando por primera vez con un proyecto completo en el que he tenido que realizar todas las tareas de planificación, análisis, diseño, implementación, pruebas y documentación.

También, me ha permitido profundizar un poco sobre los campos de la simulación de modelos y la diagnosis de fallos, que prácticamente desconocía, ya que solo sabía de su existencia por una conferencia impartida por mi tutor Aníbal Bregón en la Universidad.

Finalmente, este trabajo me ha servido para afrontar mejor un futuro laboral.

8.2. Trabajo futuro

El entorno integrado para la simulación de modelos con aplicación a la diagnosis que se ha implementado en la realización de este TFG se podría ampliar para aumentar su funcionalidad. Las ampliaciones que se podrían hacer serían dos principalmente:

- Añadir la funcionalidad de identificar fallos. Esta funcionalidad podría realizarse facilitando al usuario el poder cargar un algoritmo de identificación de fallos, la aplicación debería hacer la identificación y posteriormente debería mostrar los resultados.

- Añadir la funcionalidad de ejecutar experimentos de prognosis[1]. Para ello, la aplicación debería permitir al usuario cargar como un plug-in el algoritmo de prognosis, llevar a cabo el experimento de prognosis y por último debería mostrar los resultados.

[1] La prognosis trata de determinar cuál será el comportamiento futuro de un sistema en el que ha ocurrido un fallo y debe determinar cuánto tiempo va a poder seguir funcionando el sistema, decidiendo si puede seguir realizando su tarea correctamente en presencia del fallo o por el contrario debe abortar la tarea.

Capítulo 9
Bibliografía

[Bregón, 2008] Aníbal Bregón, Belarmino Pulido, Carlos Alonso-González. *Mejorando la Robustez en el Diagnóstico Basado en Consistencia con Posibles Conflictos.* X Jornadas de ARCA. Sistemas Cualitativos y Diagnosis, Robótica, Sistemas Domóticos y Computación. Tenerife, España,2008.

[Bregón, 2007] Aníbal Bregón, Diego García, Oscar Prieto, Belarmino Pulido, Carlos Alonso. *Sistema Reconfigurable para la Simulación en Bloque de Experimentos Útiles en Tareas de Soporte para Supervisión y Diagnosis.* Congreso Español de Informática, CEDI 2007. Zaragoza, España, 2007.

[Bregón, 2014] Aníbal Bregón. *Revisión de Técnicas de Diagnóstico de Fallos para Sistemas Dinámicos.* Sego-Bit: Revista de la Escuela de Ingeniería Informática de Segovia. Segovia, España. Número 0, Diciembre de 2014.

[Balakrishnan y Honavar, 1998] K. Balakrishnan, y V. Honavar, (1998). Intelligent diagnosis systems. *Journal of Intelligent Systems*, 8(3/4), 239-290.

[Gertler, 1998] Gertler, J. (1998). *Fault detection and diagnosis in engineering systems.* Marcel Dekker, Inc.

[Piattini *et al.*, 2004]Mario G. Piattini, José A. Calvo-Manzano, Joaquín Cervera, Luis Fernández. *Análisis y Diseño de Aplicaciones Informáticas de Gestión "Una perspectiva de Ingeniería del Software"*. Ra-Ma, 2004.

[Moldes, 2008] F. Javier Moldes Teo. *Java SE 6.* ANAYA MULTIMEDIA, 2008.

Diagnosis de fallos:

[1] http://www.dicyt.com/noticias/desarrollan-un-innovador-metodo-para-el-diagnostico-de-fallos-en-sistemas-industriales

Último acceso: 3/01/2015

Java SE:

[2] http://www.oracle.com/

Último acceso: 27/12/2014

[3] http://www.oracle.com/technetwork/es/java/javase/downloads/jdk7-downloads-1880260.html

Último acceso: 9/01/2015

Sitio web de Google para desarrolladores con información sobre matlabcontrol:

[4] https://code.google.com/p/matlabcontrol/

Último acceso: 19/12/2014

[5] https://code.google.com/p/matlabcontrol/wiki/Walkthrough

Último acceso: 19/12/2014

Apéndices

Apéndice A
Contenido del CD-ROM

El CD-ROM que se entrega junto a este documento contiene:

- Directorio llamado Documentación, en el que se encuentra este documento en formato PDF.

- Directorio llamado Aplicación, en el que se encuentra la carpeta FuncionInstante y la carpeta Ejecutable que contiene el ejecutable de la aplicación (setup_ESD.jar) y las librerías necesarias para ésta.

- Directorio llamado Código, donde se encuentra todo el código fuente de la aplicación.

- Directorio llamado Diagramas de secuencia, en el que se encuentran todos los diagramas de secuencia en formato JPG.

- Directorio llamado Sistema3tanques, donde se encuentran todos los ficheros necesarios para ejecutar un experimento de simulación y un experimento de diagnosis.

Apéndice B
Glosario

API (*Application Programming Interface*)

Interfaz de programación de aplicaciones es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.

Applet

Componente de una aplicación escrito en el lenguaje de programación Java que se ejecuta en un navegador web utilizando la Máquina Virtual de Java o en el AppletViewer de Java SE.

Bucle

En programación, es una sentencia que se realiza repetidas veces hasta que la condición asignada a dicho bucle deje de cumplirse.

Compilador

Programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar.

Depurador

Programa usado para probar y eliminar los errores del código de otros programas que están siendo implementados.

Desensamblador

Programa que se encarga de traducir un fichero escrito en código máquina a un fichero escrito en lenguaje ensamblador.

Distancia euclídea

Distancia entre dos puntos dentro de un espacio euclídeo. Esta distancia se deduce a partir del teorema de Pitágoras.

Ensamblador

Programa que se encarga de traducir un fichero fuente escrito en un lenguaje ensamblador a un fichero objeto que contiene código máquina, ejecutable directamente por el microprocesador.

Intérprete

Programa capaz de analizar y ejecutar otros programas. Realizan la traducción a medida que sea necesaria, instrucción por instrucción, y no guardan el resultado de dicha traducción.

JAR

Tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java.

RQF

Requisito funcional define una función del sistema de software o de sus componentes.

RQNF

Requisito no funcional es un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema.

UC

Caso de uso. Sus siglas proceden de su forma escrita en inglés *User Case.*