ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

SIMULACIÓN DE VENTILACIÓN MECÁNICA NO INVASIVA

Alumno: Fco. Daniel Ventura Morales

Tutora: Edurne Barrenechea Tartas

Pamplona, junio 2010

Agradecimientos

A mi tutora Edurne Barrenechea y a mi profesor Miguel Pagola por su ayuda y consejo.

Al doctor Juan Pedro Tirapu León por su ayuda e instrucción sobre ventilación mecánica.

A mi familia, por haberme dado fuerzas.

A mis amigos y compañeros de universidad por su apoyo.

A mi compañero de trabajo Santiago Gámez por su diseño de background.

Índice

Índice de figuras

Capítulo 1. INTRODUCCIÓN

El uso de la informática como herramienta de ayuda a la medicina es una realidad en auge. Desde tiempos antiguos muchas de las actividades humanas se han basado en la repetición de actos o cálculos y del mismo modo que se inventaron operaciones matemáticas básicas para simplificarlas, surgió la necesidad de mejorar las limitadas prestaciones que ofrece la mente del hombre para calcular, a medida que las diversas ciencias se hicieron más complejas. Uno de los aspectos más importantes en los que se aplica la informática en medicina es la enseñanza, y una de las técnicas usadas como apoyo a ésta es la simulación.

Conforme al diccionario de la Real Academia Española, “simular es representar algo, fingiendo o imitando lo que no es”. La simulación es la representación de un proceso o fenómeno mediante otro más simple, que permite analizar sus características; pero la simulación no es solo eso, también es algo muy cotidiano, hoy en día, puede ser desde la simulación de un examen que le hace un profesor a su alumno, la producción de textiles, alimentos, juguetes, construcción de infraestructuras por medio de maquetas, recreación de físicas en un entorno virtual, hasta el entrenamiento virtual de los pilotos de combate.

El concepto de simulación nace en 1929 de mano del ingeniero estadounidense Edwin A. Link el cual logró poner en funcionamiento el primer simulador de vuelo, surgiendo a partir de la Segunda Guerra Mundial el impulso al desarrollo de esta herramienta. Desde entonces y ante el imperativo constante de aproximarse lo máximo a la realidad, la simulación ha penetrado en todo tipo de tareas realizadas por el ser humano a un ritmo de avance similar al de la tecnología más vanguardista.

Figura 1.1.1 Primer simulador de vuelo

La simulación es un método muy útil en las Ciencias Médicas, tanto cuando se utiliza con fines didácticos como evaluativos. Este método permite acelerar el proceso de aprendizaje de los estudiantes eliminando muchas de las molestias que, durante su desarrollo, se producen a los pacientes y a la organización de los servicios de salud.

La famosa frase "Errar es humano" es válida en muchos campos o aspectos, pero errar en medicina es indeseable. Para el paciente objeto del error esto es inaceptable y para los sistemas de salud una situación intolerable. Siempre existirán los errores pero se debe intentar minimizarlos durante la enseñanza pues nadie puede ni debe asumir el costo de vidas humanas, lesiones irreparables o la disminución de la calidad de vida de los pacientes.

Se calcula que hay cerca de 400.000 muertes anuales por errores médicos, lo cual es un número similar a la suma de muertes por tabaquismo, alcohol, drogas, heridas por arma de fuego y accidentes automovilísticos. Un ejemplo, la posibilidad de morir en un hospital es 10 veces mayor que en un avión.

Para evitar tales errores, los profesionales en medicina, deben estar preparados para todo tipo de situaciones adversas que se puedan encontrar, siendo la práctica la mejor forma de llevarlo a cabo. De ahí surge la gran importancia de la simulación clínica en la medicina.

Figura 1.1.2 Simulación en la enseñanza

1.1. Ventajas de la simulación clínica

La simulación proporciona un entrenamiento que brinda al alumno la oportunidad de práctica constante de destrezas psicomotrices mientras se familiariza con instrumentos y equipos y al mismo tiempo gana experiencia en el reconocimiento de problemas y en el desarrollo de toma de decisiones, así como en el perfeccionamiento de técnicas y procedimientos que pueden presentarse en casos poco frecuentes.

El manejo apropiado de una condición de emergencia es muy difícil de enseñar durante un caso real. Durante una situación de emergencia no hay tiempo para detenerse y pensar sobre el próximo paso. Las acciones y protocolos deben fluir naturalmente. La única manera de poder practicar en una emergencia sin poner en peligro a un paciente es el aprendizaje en un ambiente simulado.

Algunas de las ventajas que ofrece la simulación son:

- No existen riesgos para el paciente.
- Situaciones críticas poco comunes en las que se requiere de respuesta rápida.
- Familiarización de los estudiantes con métodos de autoevaluación y auto aprendizaje
- Estandarización de la enseñanza
- Uso del error como un medio de aprendizaje

La simulación es utilizada en diversos campos de la medicina como por ejemplo en endoscopia, urología, cirugía... y ventilación mecánica.

1.2. ¿Qué es la ventilación mecánica no invasiva?

La insuficiencia respiratoria es una de los principales motivos por el que un paciente puede precisar ingreso en la Unidad de Cuidados Intensivos (UCI). Un paciente con fallo respiratorio implica un inadecuado intercambio gaseoso, pudiendo ser de distinta etiología, requiere a menudo soporte respiratorio para mejorar el intercambio gaseoso.

Como reseña histórica, cabe decir que, el concepto de respiración artificial fue esbozado en el siglo XVI por Andreas Vesalius, pero ha sido a partir del siglo XX cuando se ha extendido como modalidad terapéutica. Los primeros aparatos creaban una presión negativa alrededor del tórax estando el paciente encerrado en un cajón, aislado del exterior e inmovilizado. A partir de 1952, gracias a los avances de biofísica (mecánica, fluida, neumática y electrónica) y la implantación de unidades de cuidados intensivos, comienzan a desarrollarse respiradores de presión positiva. En la actualidad se dispone de ventiladores con distintos programas adaptables a las necesidades y circunstancias del paciente, que permiten una mejor monitorización de los parámetros respiratorios y ocasionan el menor impacto sobre el tejido pulmonar y sistema cardiovascular.

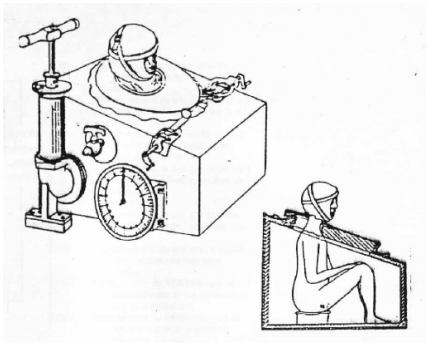


Figura 1.2.1 Primer ventilador mecánico

Figura 1.2.3 Ventilación mecánica

La Ventilación Mecánica es un procedimiento de respiración artificial que sustituye o ayuda temporalmente a la función ventilatoria de los músculos inspiratorios. Concretamente, la ventilación no invasiva es un tipo de soporte ventilatorio artificial mecánico o no mecánico que facilita la mecánica ventilatoria. Modifica el patrón ventilatorio de la insuficiencia respiratoria aguda (respiración rápida y superficial), reduciendo la frecuencia respiratoria y aumentando el volumen corriente y favoreciendo la ventilación alveolar con una presión positiva. La ventilación mecánica no invasiva permite incrementar la ventilación alveolar, manteniendo las vías respiratorias intactas. No precisa intubación endotraqueal ni traqueotomía (se aplica a través de mascarillas nasales, nasobucales o faciales), por lo que se evita el riesgo de neumonía asociada a la ventilación mecánica, disminuyendo también las necesidades de sedación del paciente. Es una ventilación más fisiológica, menos agresiva, permitiendo en algunas situaciones la alimentación oral.

Figura 1.2.2 Ventilación mecánica no invasiva

Actualmente, este tipo de ventilación se está utilizando cada vez más en las UCI debido a su fácil y rápida aplicación,

proporcionando una mayor flexibilidad y confort tanto para el paciente como para el personal sanitario

1.3. El sistema respiratorio

Antes de continuar, es necesario explicar brevemente los conceptos más importantes de la fisiología del sistema respiratorio, concretamente los que afectan de manera directa a la ventilación mecánica. Además, para poder entender los distintos tipos de ventiladores disponibles en el mercado es necesario conocer el funcionamiento del sistema respiratorio y el modelo matemático en el que se basa.

1.3.1. Conceptos

Volumen corriente o volumen tidal: Es la cantidad de aire inspirado o espirado durante cada ciclo respiratorio. Este parámetro es variable en función del tiempo y es el parámetro principal en los respiradores de volumen. La unidad de medida del volumen es mililitros aunque también se suele expresar en litros.

Presión: Es la fuerza por unidad de superficie necesaria para desplazar un volumen corriente. Se mide en centímetros de H2O. La presión se puede subdividir en diferentes presiones según los parámetros que la provocan. Para la simulación de la ventilación mecánica, interesan la presión ventilatoria (generada por el respirador) y la presión muscular (generada por el paciente). Además, según el momento de medición, obtenemos las siguientes presiones:

- **Presión pico (PIP):** Es el valor de la presión al final de la inspiración.
- **Presión meseta o plateau:** Es el valor obtenido al final de la inspiración haciendo una pausa inspiratoria, sin flujo aéreo.

Flujo: Es la cantidad de volumen que entra o sale por unidad de tiempo. Se mide en mililitros o litros por segundo o minuto.

Compliance o distensibilidad: Medición de la facilidad con la que se expanden los pulmones y el tórax durante los movimientos respiratorios (durante un cambio de volumen en relación a un cambio aplicado en la presión). Una compliance elevada indica falta de recuperación elástica de los pulmones y una baja compliance supone que es necesaria una mayor presión para producir cambios de volumen. La compliance se mide en mililitros o litros por cm de H2O. La inversa de la compliance es la elastancia.

Resistencia: Se compone de la resistencia de la vía aérea y la resistencia pulmonar. Es la fuerza conjunta de la dos anteriores que se opone al flujo aéreo. Se mide en cm H2O/L/s. A mayor resistencia se debe aplicar una mayor presión para aumentar el flujo y el volumen. Una menor resistencia implica una entrada mayor de volumen.

PEEP (Positive Airway End Pressure): La presión positiva al final de la espiración es un parámetro ampliamente utilizado durante la ventilación mecánica con fines terapéuticos. Consiste en mantener un nivel de presión positiva durante la fase

espiratoria del paciente con el fin de aumentar la capacidad residual funcional y por consiguiente aumentar el volumen pulmonar, al relajar los alveolos, por lo que produce una mejoría de la presión arterial de oxígeno (PaO2)**.**

Frecuencia respiratoria: son el número de ciclos respiratorios (inspiración + espiración) por una unidad de tiempo. Se mide en ciclos por minuto.

Tiempo inspiratorio: es el tiempo que dura la inspiración.

Tiempo espiratorio: es el tiempo que dura la espiración.

Relación Inspiración/Espiración (I:E): es la fracción de tiempo de cada ciclo dedicada a la inspiración y a la espiración. Suele fijarse en 1:2. Esto significa que si se divide en tres partes el tiempo que dura un ciclo respiratorio, dos partes la ocuparía la espiración y una la inspiración.

Pausa inspiratoria: es un intervalo de tiempo que se aplica al final de la inspiración. Cesado el flujo aéreo y cerrada la válvula espiratoria, permite distribuir el aire en el pulmón. Se da cuando el respirador ya ha concluido el tiempo inspiratorio establecido pero el paciente aún no ha comenzado la espiración.

Constante de tiempo: Es el producto de la resistencia y la compliance. Se mide en segundos. Se le denomina como una constante porque para cualquier valor de R y C, la constante de tiempo es siempre igual al tiempo necesario para vaciar los pulmones en un 63%. Por ejemplo, cuando el tiempo espiratorio en un ventilador es igual a la constante de tiempo, el paciente habrá exhalado pasivamente sólo un 63% del volumen corriente cuando se inicie la siguiente fase inspiratoria.

Trigger o disparador: Es la sensibilidad. Se programa cuando se emplean modos de ventilación asistida. Determina la presión o flujo negativo requerido para iniciar una respiración mecánica. Si es pequeña, el respirador funcionará en modo controlado.

Rampa: Es el tiempo que tarda la presión en alcanzar el pico en los ventiladores controlados por presión. Se mide en segundos.

1.3.2. Ecuación del movimiento del sistema respiratorio

El sistema respiratorio se basa en un modelo matemático que tiene como base la llamada ecuación del movimiento:

Presión = Volumen/Compliance + Resistencia x Flujo + PEEP

En esta ecuación la resistencia y la compliance son generalmente constantes (inherentes al paciente). La presión, el flujo y el volumen cambian durante el tiempo, es decir, son las variables que pueden ser modificadas por el ventilador. De la ecuación de movimiento se desprende que un ventilador puede modificar una de las tres variables.

1.3.3. Variables de control

Las variables de control son las variables de que manipula el ventilador para causar inspiración según su modo de funcionamiento. Estas variables pueden ser: volumen, presión o flujo, y son representadas mediante curvas que el ventilador genera a partir de su medición. Las curvas originadas pueden tener cuatro formas diferentes: rectangular, exponencial (creciente o decreciente), sinusoidal o rampa (ascendente o descendente).

De acuerdo con la ecuación del movimiento del sistema respiratorio, si la presión ventilatoria es la variable de control, entonces el volumen y el flujo son dependientes de la resistencia, la compliance y la presión muscular. Si el volumen o el flujo son las variables de control, entonces la presión ventilatoria es dependiente de la resistencia, la compliance y la presión muscular. Solo puede manipularse una variable a la vez y ésta es la que provoca la variación de los restantes parámetros durante el ciclo respiratorio.

1.3.4. Fases del ciclo respiratorio

El ciclo respiratorio consta de cuatro fases:

- Disparo o inicio de la inspiración.
- Transcurso de la inspiración.
- Ciclado o inicio de la espiración.
- Transcurso de la espiración.

La duración de cada fase y el cambio entre una y otra depende de la frecuencia respiratoria y de alguna de las siguientes variables: volumen, presión, flujo o tiempo. Por ejemplo, el ciclado puede ocurrir al pasar un tiempo determinado o cuando se alcanza un determinado volumen.

1.4. Clasificación de los respiradores

Existen una gran variedad de modelos de ventilación mecánica definidos por la combinación de las diferentes formas de funcionamiento. A continuación se detallan las clasificaciones más importantes.

1.4.1. Clasificación por variable de control

Una primera clasificación de los ventiladores se basa en el control que hacen sobre los tres parámetros variables del paciente, es decir, las variables que sufren modificaciones respecto al tiempo (no estáticas). Estos parámetros son el volumen, la presión y el flujo. El ventilador produce la inspiración al manipular una variable, la cual se mantiene constante sin importar si cambian las condiciones del pulmón. Un ventilador sólo puede modificar una de las variables al mismo tiempo, por lo que se clasifican en:

- **Controladores de presión:** En la ecuación de movimiento del sistema respiratorio, la presión es determinada por variaciones en el flujo y en el volumen. Son máquinas que permiten un flujo de gas al disparar una válvula accionada por el esfuerzo del paciente o por un mecanismo automático, el cual se mantiene hasta que se alcanza una presión máxima preestablecida. Seguidamente, esta presión se mantiene hasta que termina el tiempo inspiratorio.

- **Controladores de volumen:** Con estos ventiladores el volumen se mantiene inalterado a medida que se cambia la resistencia. El flujo también se mantiene constante. De acuerdo a la ecuación de movimiento del sistema respiratorio, si el volumen es la variable controlada y el flujo se mantiene constante debido a su relación con el volumen, la presión debe ser la variable cambiante en presencia a una carga a vencer, que también cambia con el tiempo.

- **Controladores de flujo:** Mantienen un flujo y volumen constantes ante la presencia de una carga variable (resistencia). La diferencia fundamental con los de volumen es que lo miden en forma indirecta con un transductor de flujo. Este es medido y calculado como una función del tiempo.

Es importante no confundir el modo según la variable de control con el modo "variable" control. Por ejemplo, es diferente un ventilador controlado por presión que un ventilador de presión control.

1.4.2. Clasificación por ciclado

Los ventiladores, además, se clasifican en función del mecanismo de ciclado (*ciclado:* sistema por el que cesa la inspiración y se inicia la fase inspiratoria pasiva):

- **Ciclados por presión:** Ciclados por tiempo para iniciar y limitar el ciclo inspiratorio de la ventilación y limitados por presión para controlar el flujo y el volumen de cada respiración. El volumen recibido por el paciente y el tiempo de la inspiración, están en función de la resistencia de la vía aérea y de la compliance pulmonar.

- **Ciclados por volumen:** En estos equipos el factor determinante del paso de inspiración a expiración es el volumen prefijado. El control del volumen inspiratorio está relacionado con el regulador de flujo inspiratorio; mientras aumenta el volumen, el flujo es constante, y disminuye a cero en la pausa inspiratoria (no existe cambio en el volumen corriente).

- **Ciclados por tiempo:** se mantiene constante el tiempo inspiratorio, variando por tanto el volumen que se entrega y la presión que se genera.

- **Ciclados por flujo:** La inspiración termina cuando el flujo inspiratorio disminuye por debajo de un nivel predeterminado, con independencia del volumen, tiempo o presión generada. Su inconveniente es que pueden no entregarse volúmenes suficientes y no alcanzar frecuencias respiratorias adecuadas.

1.4.3. Clasificación por modalidad respiratoria

Los ventiladores se clasifican también según influya y se permita el esfuerzo del paciente:

- **Ventilación espontánea:** se basa en la demanda del paciente. El flujo y el volumen están determinados por el esfuerzo inspiratorio del individuo. El flujo se inicia cuando el esfuerzo inspiratorio alcanza el nivel de sensibilidad preestablecido. A mayor esfuerzo inspiratorio mayor será el flujo obtenido. El disparo, el limite y el ciclado son dados por el paciente (paciente hace todo el trabajo)

- **Ventilación asistida:** se inicia cuando el esfuerzo inspiratorio del paciente es igual al nivel de sensibilidad del ventilador. Disparada por el paciente, pero limitada y ciclada por el ventilador.

- **Ventilación controlada:** el paciente es ventilado de acuerdo a las variables de control preestablecidas en el respirador. En ausencia de un esfuerzo inspiratorio del paciente, el ventilador proporciona la respiración controlada. Disparada, limitada y ciclada por el ventilador, (todo el trabajo es realizado por éste).

Capítulo 2. PRESENTACIÓN DEL PROYECTO

2.1. Descripción del problema

El principal problema que existe a la hora del aprendizaje, por parte del alumno en cuanto al uso de un aparato de ventilación mecánica, es la práctica.

Como bien se puede imaginar, no se puede utilizar un aparato de ventilación mecánica en una persona sana, o al contrario, un alumno no puede realizar prácticas sobre un paciente en el que su vida depende del correcto uso del ventilador.

Existen aparatos o "muñecos" que se pueden conectar a un respirador y que simulan el aparato respiratorio humano pero, debido al elevado coste de los respiradores es inviable para un centro el poder disponer de uno por alumno.

El propósito de este proyecto es realizar una aplicación informática para solucionar los problemas anteriores. Por ello, el objetivo del sistema es simular la utilización práctica de un aparato de ventilación mecánica sobre una persona con insuficiencia respiratoria. El software por tanto debe simular la mecánica respiratoria de un paciente y la interacción que produce sobre ésta el uso de una máquina de respiración artificial, abarcando diferentes modos de ventilación existentes.

2.2. Ámbito

El principal uso del software se realizará en el área de la educación. La aplicación será utilizada tanto por profesores para apoyo a sus explicaciones, como por alumnos para realización de prácticas y como recurso didáctico.

El sistema se denomina VMSof, y consiste en un único programa que realiza las funciones necesarias para la simulación. En particular, debe permitir las siguientes funciones de configuración:

- Asignar valores a los parámetros respiratorios inherentes al paciente.

- Elegir la modalidad ventilatoria adecuada mediante la definición de diferentes aspectos del respirador.
- Asignar valores a los parámetros del respirador.

La interfaz gráfica facilita la visión de los resultados en tiempo real, obtenidos de la simulación, mediante gráficas con funciones especiales.

El sistema no necesita de una base de datos, siendo posible almacenar las configuraciones en ficheros con extensión .pvm propia del software.

2.3. Objetivos generales del proyecto

- Estudiar, analizar y comprender los diferentes datos clínicos a manejar. Estos datos comprenden la fisiología del aparato respiratorio y el funcionamiento y manejo de cada uno de los diferentes ventiladores.
- Diseñar una interfaz gráfica que permita la entrada de datos de una forma sencilla y amigable al usuario.
- Diseñar una interfaz gráfica que permita mostrar la salida de datos mediante gráficas dinámicas en función del tiempo.
- Desarrollar las clases y algoritmos necesarios que realicen la simulación de la respiración del paciente en función de los parámetros clínicos y bioquímicos necesarios y la variación de éstos en función de los valores de los parámetros propios del ventilador.
- Realizar la documentación asociada a las diferentes fases del desarrollo de software tales como el análisis de requisitos, análisis de diseño, codificación, pruebas y manual de usuario.

2.4. Objetivos y funciones del software

La simulación se realiza en tiempo real mediante la configuración previa (manual o por defecto) de las diferentes variables del paciente y del respirador por parte del usuario. Estas variables tomarán valores en un rango determinado, siendo este acorde a la realidad.

El usuario debe disponer de algunas facilidades para la observación de los resultados, siendo estos mostrados en formato texto o en formato gráfico. Entre estas facilidades se encuentra la posibilidad de hacer zoom sobre un área determinada de la gráfica o pausarla en un determinado momento en el tiempo.

La simulación del respirador no se basa en un modelo en concreto si no que admite diferentes configuraciones disponibles en aparatos comerciales, pudiendo “construir” el respirador que mejor convenga en una determinada situación.

El objetivo de la aplicación es conseguir una simulación realista de casos prácticos en ventilación mecánica totalmente configurables por el usuario. Las funciones principales del usuario son:

- Editar los parámetros respiratorios inherentes al paciente.
- Editar los parámetros del respirador.
- Almacenar o recuperar configuraciones personales.
- Interactuar con las gráficas de valores.

2.5. Herramientas y técnicas utilizadas

La aplicación ha sido desarrollada en el lenguaje JAVA. Se ha elegido dicho lenguaje, en primer lugar, porque se ha considerado más conveniente usar un lenguaje orientado a objetos a un lenguaje estructural. Además, Java, frente a otros lenguajes orientados a objetos, es muy utilizado en todo tipo de proyectos, su sintaxis es sencilla y también ofrece independencia de la plataforma.

Como editor de programación se ha utilizado la herramienta NetBeans, una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas.

En la realización del proyecto se han usado diversas librerías y técnicas como son la librería Substance que modifica el "look and feel" de la aplicación o la librería JfreeChart que permite la representación de datos en diferentes tipos de gráficas.

Se ha utilizado la herramienta Izpack (formato .jar) junto con Launch4j (formato .exe), ambas open source, para realizar los ejecutables de instalación de la aplicación.

Para el desarrollo y ejecución de la aplicación se ha utilizado un ordenador con el sistema operativo Windows XP, la máquina virtual de Java (Java SE 6) y el Java SE Development Kit (JDK 6).

La documentación del código ha sido escrita en formato JavaDoc y utilizando un plugin de NetBeans se ha generado el documento html con la API y la documentación completa del código. Utilizando otro plugin en NetBeans se han realizado también los diagramas de casos de uso, de secuencia y de clases de la aplicación.

Por último, este documento ha sido realizado con MS Word 2007.

Capítulo 3. REQUISITOS DEL SOFTWARE

En la elaboración del proyecto se han tenido en cuenta los requisitos impuestos por los usuarios finales de la aplicación. Dichos usuarios finales son médicos, profesores y estudiantes.

El Doctor Juan Pedro Tirapu León trabaja en la Unidad de Cuidados Intensivos del Hospital de Navarra y a su vez imparte cursos de formación a sus compañeros. Él ha sido la persona de contacto que ha marcado los requisitos necesarios, tanto de apariencia (no funcionales) como funcionales. Estos requisitos fueron obtenidos tras múltiples reuniones con el Doctor J. P. Tirapu, tanto en los laboratorios de la UPNA como en la UCI del Hospital de Navarra, y están basados en el aspecto y funcionalidad de los respiradores utilizados en la UCI y en el uso que se dará al nuevo software.

El proyecto consiste en una aplicación que simula la interacción respirador-paciente en diferentes casos clínicos configurables. Aunque exista más de un usuario final, el rol del usuario en la aplicación es siempre el mismo. Por ello, la aplicación no gestiona diferentes tipos de usuarios ya que es indiferente que la aplicación esté siendo utilizada por un profesor o por un alumno.

Todas las funcionalidades implementadas por la aplicación son comunes a los diferentes usuarios finales por lo que todas se pueden agrupar bajo las necesidades de un único rol de usuario común.

3.1. Requisitos Hardware

Los requisitos hardware vienen impuestos principalmente por el lugar donde se implanta la aplicación. En este caso, la aplicación podrá ser instalada en cualquier ordenador de sobremesa o portátil con un hardware básico.

3.2. Requisitos Software

La aplicación ha de ser una herramienta que dé soporte a todas las necesidades que surjan del análisis y se adapte a la realidad de la simulación.

El usuario, cuando espera algo de un sistema, describe en lenguaje natural sus expectativas. Estas expectativas pueden traducirse en lo que llamamos requerimientos del sistema o requerimientos funcionales.

Además, en algunos casos, durante la vida del sistema, algunos clientes, usuarios, analistas y desarrolladores aportan nuevas ideas que deben plasmarse en requerimientos reales.

3.2.1. Requisitos no funcionales

Los requerimientos no funcionales especifican propiedades del sistema no relacionados de manera explícita con la funcionalidad deseada como pueden ser restricciones de entorno e implementación, rendimiento, dependencias de plataforma, mantenimiento, etc.

R.1.- La aplicación debe funcionar correctamente en entornos Windows, siendo deseable la portabilidad a diferentes plataformas (Linux, Mac OS X…)

R.2.- La aplicación debe tener un aspecto profesional y similar a la interfaz de respiradores modernos como el Evita.

R.3.- En el menú de configuración, los parámetros del paciente y del respirador deben estar claramente diferenciados y organizados para mayor facilidad y comodidad.

R.4.- La aplicación debe tener una interfaz suficientemente intuitiva para que el personal especializado en medicina pueda entender su funcionamiento sin necesidad de ayuda externa.

R.5.- Se debe mostrar un logotipo del Hospital de Navarra o del servicio de salud, quedando reflejada la entidad a la que va destinado el software.

R.6.- Una vez comenzada la simulación se debe poder mostrar simultáneamente las tres gráficas principales donde se reflejan los valores de la presión ventilatoria, el volumen tidal y el flujo respiratorio.

R.7.- La aplicación debe tener una arquitectura modular que facilite una rápida implementación e integración de nuevos tipos de respiradores.

R.8.- Ha de emplearse plataformas de desarrollo de libre distribución, anulando costes de licencias.

R.9.- La instalación del software debe realizarse de forma sencilla mediante la ejecución de un fichero .exe para el caso de Windows.

R.9.- El software debe estar ampliamente documentado

3.2.2. Requisitos Funcionales

Son aquellos requisitos que definen el funcionamiento de la aplicación.

3.2.2.1. Requisitos de la simulación médica

R.S.M.1.- El modelo matemático a utilizar para la simulación de la respiración de un ser humano es el basado en la ecuación del movimiento del sistema respiratorio.

R.S.M.2.- Se debe simular el funcionamiento de una máquina de gases básica.

R.S.M.1.- En la simulación, se deben poder configurar los principales parámetros inherentes al paciente que participan en la respiración: presión muscular, compliance pulmonar, resistencia pulmonar y frecuencia respiratoria.

R.S.M.1.1.- La presión muscular del paciente debe estar medida en cm H2O y su valor debe pertenecer al rango (0-70) tomando como valor por defecto 5 cm H2O.

R.S.M.1.2.- La compliance pulmonar se mide en ml/cm H2O y su valor debe pertenecer al rango (10-250) tomando como valor por defecto 70 ml/cm H2O.

R.S.M.1.3.- La resistencia pulmonar es medida en cm H2O/L/s y su valor debe pertenecer al rango (1-50) tomando como valor por defecto 5 cm H2O/L/s.

R.S.M.1.4.- La frecuencia respiratoria se debe medir en respiraciones por minuto y su valor debe pertenecer al rango (1-50) tomando como valor por defecto 12 rpm.

R.S.M.2.- Para la simulación de la máquina de gases (equilibrio ácido-base), los parámetros necesarios son el pH del paciente, el pCO2 y la pO2. A partir de dichos parámetros se debe calcular el pH esperado que debería tener el paciente y el valor del equilibrio ácido-base del paciente.

R.S.M.2.1.- El valor del pH debe pertenecer al rango (6,9-7,7), tomando como valor por defecto 7,4.

R.S.M.2.2.- La pCO2 se mide en mm Hg y su valor debe pertenecer al rango (10-120) tomando 40 mm Hg como valor por defecto.

R.S.M.2.3.- La pO2 se mide en mm Hg y su valor debe pertenecer al rango (20-700) tomando 95 mm Hg como valor por defecto.

R.S.M.3.- Los parámetros a controlar del respirador, comunes a todas las combinaciones de respiradores, son la variable de control (presión o volumen), la PEEP (Positive End Expiratory Pressure) y la frecuencia respiratoria.

R.S.M.3.1- En los respiradores de presión, la variable de control es la presión ventilatoria, la cual se mide en cm H2O y toma valores en el rango (0-100). Por defecto, el respirador comienza a funcionar con una presión de 20 cm H2O.

R.S.M.3.2.- En los respiradores de volumen, la variable de control es el volumen corriente, el cual se mide en ml y toma valores en el rango (200-7000). Por defecto, el respirador comienza a funcionar con un volumen de 500 ml.

R.S.M.3.3.- La PEEP se mide en cm H2O y toma valores en el rango (0-10) siendo 0 cm H2O el valor por defecto.

R.S.M.3.4.- La frecuencia respiratoria del respirador toma los mismos valores que la frecuencia respiratoria del paciente (1-50) y su valor por defecto es 12 rpm.

R.S.M.4.- Se deben implementar las modalidades asistida y controlada de interacción del respirador con el paciente. Si existe esfuerzo inspiratorio por parte del paciente la modalidad deberá ser asistida y si no existe esfuerzo será controlada. La aplicación debe permitir el paso de una a otra modalidad de forma automática según las condiciones del paciente sin la intervención del usuario.

R.S.M.5.- La aplicación debe simular el funcionamiento tanto de los respiradores de presión como de los respiradores de volumen.

R.S.M.6.- Se debe dar la opción de elegir el método de ciclado para el paso de la fase inspiratoria a la fase espiratoria. Las opciones de ciclado son: por tiempo inspiratorio, por relación I:E (tiempo), por volumen y por presión.

R.S.M.7.- Cada tipo de respirador deberá tener la posibilidad de elegir únicamente los métodos de ciclado que estén disponibles para su categoría.

R.S.M.7.1.- Los respiradores controlados por presión admiten los métodos de ciclado por tiempo inspiratorio, por relación I:E (tiempo) y por volumen.

R.S.M.7.2.- Los respiradores controlados por volumen admiten los métodos de ciclado por tiempo inspiratorio, por relación I:E (tiempo) y por presión.

R.S.M.7.3.- En un respirador ciclado por tiempo inspiratorio, dicho tiempo nunca puede ser inferior a 0,1s ni mayor o igual que el tiempo que tarda el paciente en realizar una respiración completa. Es decir, siempre debe haber un tiempo espiratorio.

R.S.M.7.4.- En un respirador ciclado por tiempo empleando la relación I:E, el factor I (inspiración) nunca debe ser mayor que el factor E (espiración) ni menor que 1.

R.S.M.7.5.- En los respiradores ciclados por volumen o presión, si no se alcanza el valor de ciclado antes de un tiempo inspiratorio marcado por la relación I:E, comienza la fase espiratoria.

R.S.M.8.- El respirador constará de alarmas para determinadas variables impidiendo que el parámetro limitado sobrepase el valor indicado. Las variables limitadas son la presión ventilatoria y el volumen corriente.

R.S.M.8.1.- En los respiradores controlados por presión, las opciones de alarma son: apagada y alarma de volumen.

R.S.M.8.2.- En los respiradores controlados por volumen, las opciones de alarma son: apagada y alarma de presión.

3.2.2.2. Requisitos de la interfaz de usuario

R.I.U.1- La aplicación se basará en una serie de menús o formularios gráficos heredados de las clases del paquete SWING de Java.

R.I.U.2.- Se facilitará una interfaz donde el usuario pueda elegir si desea comenzar una sesión con una nueva configuración o elegir una configuración almacenada previamente.

R**.I.U.3.-** Existirá un único formulario donde se encontrarán todas las opciones de configuración disponibles.

R.I.U.4.- Los parámetros se encontrarán organizados según su función de forma que facilite al usuario la configuración de la simulación de forma intuitiva.

R.I.U.5.- Todos los parámetros numéricos se introducirán mediante jTextbox o jSpinner delimitando siempre el rango de valores para cada parámetro en concreto.

R.I.U.6.- Para la configuración del tipo de respirador se debe utilizar jRadioButtons.

R.I.U.7.- En todo momento permanecerán deshabilitadas las opciones que no estén disponibles para un determinado tipo de respirador. Por ejemplo, si el respirador es controlado por presión, se deshabilitará el modo de ciclado por presión y la alarma de presión.

R.I.U.8.- Los parámetros de configuración del paciente y del respirador deberán estar disponibles durante la simulación y permitir cambios en los valores por parte del usuario.

R.I.U.9.- En el menú de simulación se mostrarán los datos correspondientes al tipo de respirador elegido, la presión ventilatoria pico, el flujo respiratorio pico y el volumen corriente pico en formato texto.

R.I.U.10.- Los valores de la presión ventilatoria, el flujo respiratorio y el volumen corriente se deben mostrar en gráficas dinámicas en función del tiempo.

R.I.U.11.- El usuario debe poder elegir los datos que desea visualizar en cada gráfica.

R.I.U.12.- Las gráficas podrán representar las variables de control de forma individual o simultáneamente en combinaciones de a dos.

R.I.U.13.- En todo momento las gráficas estarán contenidas en la aplicación de forma que se mantenga un aspecto correcto de la interfaz.

R.I.U.14.- La gráfica se actualizará a una velocidad acorde a la velocidad a la que se actualizan las variables del paciente de manera que muestre los valores correctos en todo momento.

R.I.U.15.- La gráfica debe implementar las siguientes funcionalidades:

- Debe permitir la ampliación de la gráfica al tamaño completo de la aplicación.
- Debe permitir la opción de pausa.
- Debe permitir realizar zoom en una zona concreta de la gráfica.

R.I.U.16.- Desde el menú de simulación se debe poder regresar al menú de inicio, cargar una configuración existente, ver la ayuda y salir de la aplicación.

R.I.U.17.- La aplicación debe adaptarse a los diferentes modos y resoluciones de pantalla de forma que se mantengan las proporciones de la interfaz y el aspecto de la aplicación no se vea alterado.

3.2.2.3. Requisitos de almacenamiento

R.A.1.- La aplicación debe ser capaz de almacenar y recuperar los datos de un determinado paciente junto con la configuración del respirador

R.A.2.- Los datos han de ser almacenados de acuerdo al formato propio de la aplicación.

R.A.3.- El formato a manejar es el .pvm, creado exclusivamente para dicha aplicación, y que contiene los objetos referentes al paciente y al respirador.

R.A.4.- El usuario debe poder almacenar la configuración del paciente y del respirador antes y durante el transcurso de la simulación.

R.A.5.- El fichero debe ser almacenado en el directorio por defecto de la aplicación.

R.A.6.- Si existiese un fichero con el mismo nombre se debe dar la opción de reemplazarlo.

R.A.7.- El usuario debe poder elegir el fichero con el paciente deseado de entre el árbol de directorios del sistema operativo, habiendo en todo caso un directorio por defecto con los pacientes almacenados anteriormente con la aplicación.

R.A.8.- La aplicación debe comprobar que los datos de entrada están en el formato adecuado. En caso de existir errores sintácticos, estos deben ser reconocidos.

R.A.9.- En caso de que los datos no se almacenen íntegros o exista un problema de escritura, hay que detectarlo y notificarlo al usuario.

Capítulo 4. ANÁLISIS DE REQUISITOS

En este capítulo se realiza el análisis de los requisitos obtenidos en las diversas entrevistas con el Doctor J.P. Tirapu con el objeto de generar los distintos casos de uso que componen la aplicación.

Los casos de uso describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario. De este modo se definen los límites del sistema y las relaciones entre el sistema y el entorno, en nuestro caso el usuario.

El diagrama de casos de uso de la aplicación es un diagrama sencillo. La interacción del usuario con la aplicación es baja debido a la naturaleza de la aplicación. La mayor complejidad se encuentra en las interacciones entre los distintos componentes de la aplicación y los flujos de mensajes que generan, los cuales se explicarán en detalle más adelante.

La siguiente figura muestra de forma global el diagrama de casos de uso y a continuación se detalla cada uno de ellos.

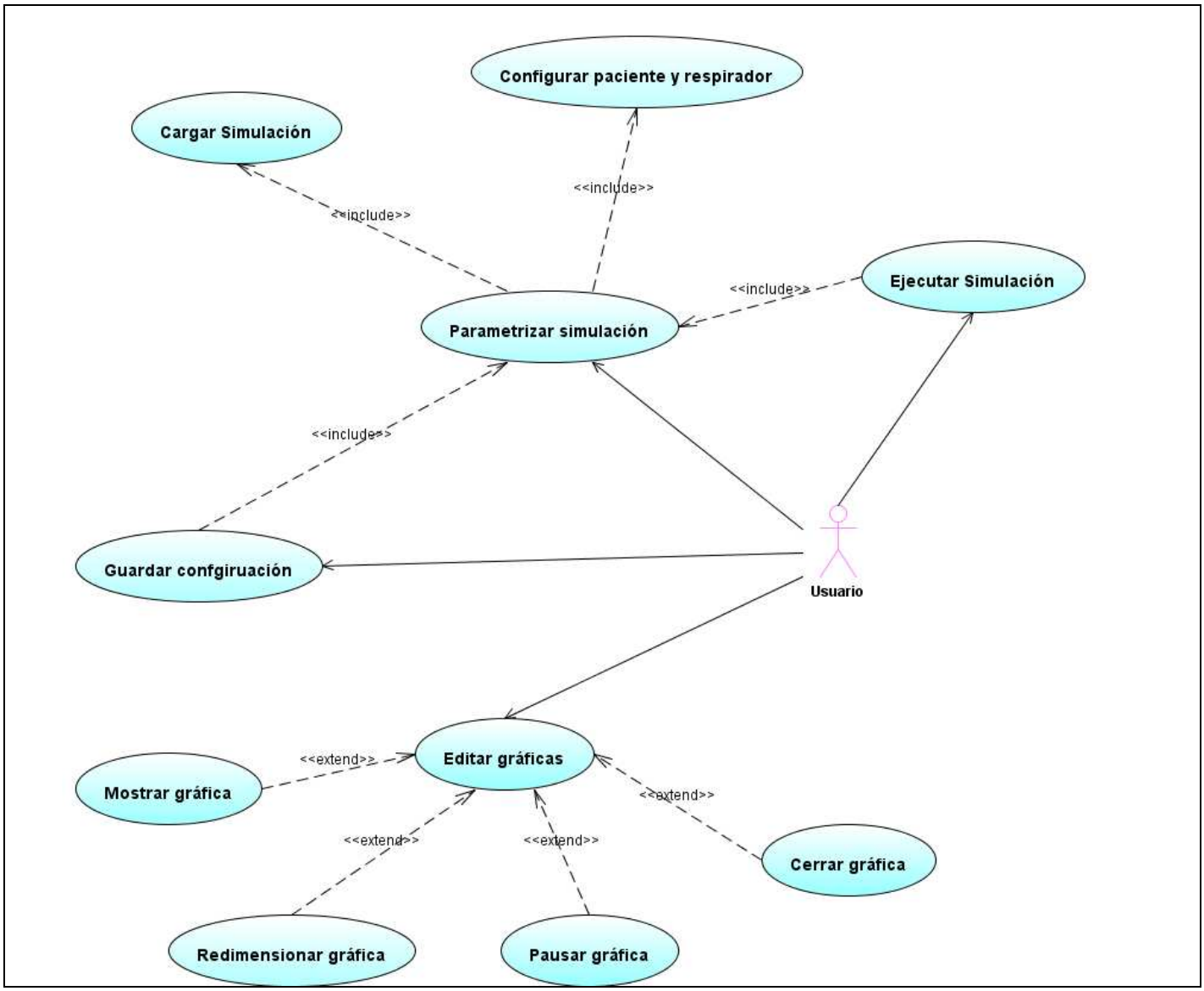


Figura 3.2.1 Diagrama de casos de uso completo

4.1. Caso de Uso: Parametrizar simulación

4.1.1. Descripción

Permite al usuario elegir el valor de los parámetros de configuración antes o durante la simulación. Para ello puede elegir cargar los valores contenidos en un fichero o elegir manualmente el valor de cada uno de los parámetros que intervienen en la simulación. Una vez que termina la configuración correctamente, se crea la simulación.

4.1.2. Actores

El actor principal es el usuario.

4.1.3. Precondición

El usuario se encuentra en el menú de inicio de la aplicación o en el menú de simulación.

4.1.4. Postcondición

Al acabar el caso de uso correctamente se accede el menú de simulación. Si el caso de uso termina en error se devuelve al usuario al menú de inicio.

4.1.5. Flujo principal

4.1.5.1. Flujo básico 1:

- Paso 1: El usuario pulsa el botón “Nueva simulación” (Menú de inicio) o “Nuevo” en la barra de menú (Menú de simulación).
- Paso 2: Se ejecuta el caso de uso Configurar paciente y respirador.
- Paso 3: Se despliega el menú de simulación.

4.1.5.2. Flujo básico 2:

- Paso 1: El usuario pulsa el botón “Cargar simulación” (Menú de inicio) o “Abrir” en la barra de menú (Menú de simulación).
- Paso 2: Se ejecuta el caso de uso Cargar simulación.
- Paso 3: Se despliega el menú de simulación.

4.1.6. Flujo alternativo

- Paso 3: El caso de uso devuelve error

- Paso 4: Se despliega el menú de inicio

4.2. Caso de Uso: Configurar paciente y respirador

4.2.1. Descripción

Permite al usuario elegir el tipo de respirador a simular y configurar manualmente el valor de todos los parámetros que participan en la simulación.

4.2.2. Actores

El actor principal es el usuario.

4.2.3. Precondición

El usuario se encuentra en el menú de configuración de la aplicación o en el menú de simulación.

4.2.4. Postcondición

Al acabar el caso de uso correctamente se configura el respirador y el paciente con los valores elegidos.

4.2.5. Flujo principal

- Paso 1: El usuario modifica el valor del parámetro elegido.
- Paso 2: El sistema comprueba que el nuevo valor sea correcto.
- Paso 3: Se actualiza el paciente o el respirador según el parámetro modificado.
- Paso 4: Vuelve al paso 1 o termina la configuración.

4.2.6. Flujo alternativo

- Paso 3: El valor elegido no es correcto, se mantiene el valor anterior.
- Paso 4: Vuelve al paso 1 o termina la configuración.

4.3. Caso de Uso: Cargar simulación

4.3.1. Descripción

Permite al usuario cargar desde fichero la configuración de una simulación ya existente.

4.3.2. Actores

El actor principal es el usuario.

4.3.3. Precondición

Su ejecución se realiza desde el caso de uso Parametrizar simulación. El usuario se encuentra en el menú de inicio o en el menú de simulación.

4.3.4. Postcondición

Al acabar el caso de uso correctamente se configura el respirador y el paciente con los valores contenidos en el fichero.

4.3.5. Flujo principal

- Paso 1: Se despliega una ventana con el árbol de directorios del sistema.
- Paso 2: El usuario elige un fichero de configuración.
- Paso 3: El sistema comprueba que el fichero sea correcto.
- Paso 4: El sistema muestra un mensaje de confirmación.
- Paso 5: Se recupera la configuración del paciente y del respirador.

4.3.6. Flujo alternativo

- Paso 4: El fichero no es válido o está corrupto, no se crea el paciente ni el respirador.
- Paso 5: El sistema muestra un mensaje de error.

4.4. Caso de Uso: Guardar configuración

4.4.1. Descripción

Permite al usuario guardar en un archivo la configuración establecida para la simulación.

4.4.2. Actores

El actor principal es el usuario.

4.4.3. Precondición

Se debe ejecutar en primer lugar el caso de uso Parametrizar simulación. El usuario se encuentra en el menú de configuración o en el menú de simulación.

4.4.4. Postcondición

Al acabar el caso de uso correctamente la información de la simulación queda almacenada en un archivo .pvm.

4.4.5. Flujo principal

- Paso 1: El usuario pulsa el botón “Guardar” (Menú de configuración) o “Guardar” en la barra de menú (Menú de simulación).
- Paso 2: Se despliega un formulario donde introducir un nombre para el fichero.
- Paso 3: El usuario escribe un nombre y pulsa el botón “Aceptar”.
- Paso 4: El sistema comprueba que el nombre sea válido.
- Paso 5: El sistema almacena el archivo.
- Paso 6: Se despliega el menú de simulación.

4.4.6. Flujo alternativo

- Existe un archivo con el mismo nombre:
  - Paso 4: El sistema pregunta si desea reemplazarlo.
  - Paso 5: El usuario elige:
    - SI: El sistema sobrescribe el archivo y despliega el menú de simulación.
    - NO: Vuelve al paso 3.

- El campo del nombre está vacío:
  - Paso 4: El sistema muestra un mensaje de error.

- Paso 5: Vuelve al paso 3.

4.5. Caso de Uso: Ejecutar simulación

4.5.1. Descripción

Permite al usuario comenzar la simulación con los valores elegidos para el paciente y el respirador.

4.5.2. Actores

El actor principal es el usuario.

4.5.3. Precondición

Se debe ejecutar en primer lugar el caso de uso Parametrizar simulación. El usuario se encuentra en el menú de simulación. La simulación debe estar inactiva.

4.5.4. Postcondición

Al acabar el caso de uso la simulación se ejecuta correctamente hasta que se ejecute alguna acción que la detenga.

4.5.5. Flujo principal

- Paso 1: El usuario pulsa el botón “Conectar” en el menú de simulación.
- Paso 2: El sistema activa el respirador.
- Paso 3: El sistema activa las gráficas.
- Paso 4: Comienza el ciclo respiratorio del paciente.
- Paso 5: Las gráficas muestran los valores de la simulación.

4.5.6. Flujo alternativo

- Existe un archivo con el mismo nombre:
  - Paso 4: El sistema pregunta si desea reemplazarlo.
  - Paso 5: El usuario elige:
    - SI: El sistema sobrescribe el archivo y despliega el menú de simulación.
    - NO: Vuelve al paso 3.

- El campo del nombre está vacío:
  - Paso 4: El sistema muestra un mensaje de error.

- Paso 5: Vuelve al paso 3.

4.6. Caso de Uso: Editar gráficas

4.6.1. Descripción

Permite al usuario realizar diferentes funciones sobre las gráficas simuladas. Estas funciones son abrir, cerrar, redimensionar y pausar.

4.6.2. Actores

El actor principal es el usuario.

4.6.3. Precondición

El usuario se encuentra en el menú de simulación. La simulación debe estar inactiva.

4.6.4. Postcondición

Ninguna.

4.6.5. Flujo principal

4.6.5.1. Escenario 1: Mostrar gráfica

- Paso 1: El usuario elige, de un menú desplegable, la gráfica que desea ver.
- Paso 2: El usuario pulsa el botón “Mostrar”.
- Paso 3: El sistema comprueba que exista una posición libre para la nueva gráfica.
- Paso 4: El sistema despliega la gráfica en el primer hueco libre que encuentra.

4.6.5.2. Escenario 2: Cerrar gráfica

- Paso 1: El usuario pulsa el botón “Cerrar” de la gráfica que desea eliminar.
- Paso 2: El sistema elimina la gráfica y libera la memoria utilizada.

4.6.5.3. Escenario 3: Redimensionar gráfica

- Paso 1: El usuario pulsa el botón “Maximizar” de la gráfica que desea ampliar.
- Paso 2: El sistema redimensiona la gráfica al tamaño completo de la ventana.
- Paso 3: El usuario pulsa el botón “Restaurar” de la gráfica.
- Paso 4: El sistema devuelve la gráfica a su tamaño original.

4.6.5.4. Escenario 4: Pausar gráfica

- Paso 1: El usuario pulsa el botón “Pausar” de la gráfica que desea detener.
- Paso 2: El sistema detiene el refresco de la gráfica.
- Paso 3: El usuario pulsa el botón “Reanudar”.
- Paso 4: El sistema borra el gráfico y activa el refresco de la gráfica.

4.6.6. Flujo alternativo

4.6.6.1. Escenario 1: Mostrar gráfica

- Paso 4: No hay espacio para más gráficas, el sistema muestra un mensaje de error.

Capítulo 5. MODELO DE ANÁLISIS

El objeto de este capítulo es realizar el análisis de los casos de usos obtenidos en el capítulo anterior.

Dadas las características de los componentes a implementar, para el análisis de los casos de uso se utilizan diagramas de interacción, en concreto, diagramas de secuencia.

Los diagramas de secuencia muestran la interacción de un conjunto de objetos o componentes en una aplicación representando los mensajes enviados entre ellos a través del tiempo.

En este apartado se muestran los diagramas de secuencia más importantes de la aplicación. Se analizarán los casos de uso descritos anteriormente, además de la interacción entre los elementos que componen el núcleo de la simulación.

5.1. Diagrama de secuencia: Cargar simulación

La figura 5.1 muestra el diagrama de secuencia Cargar simulación.

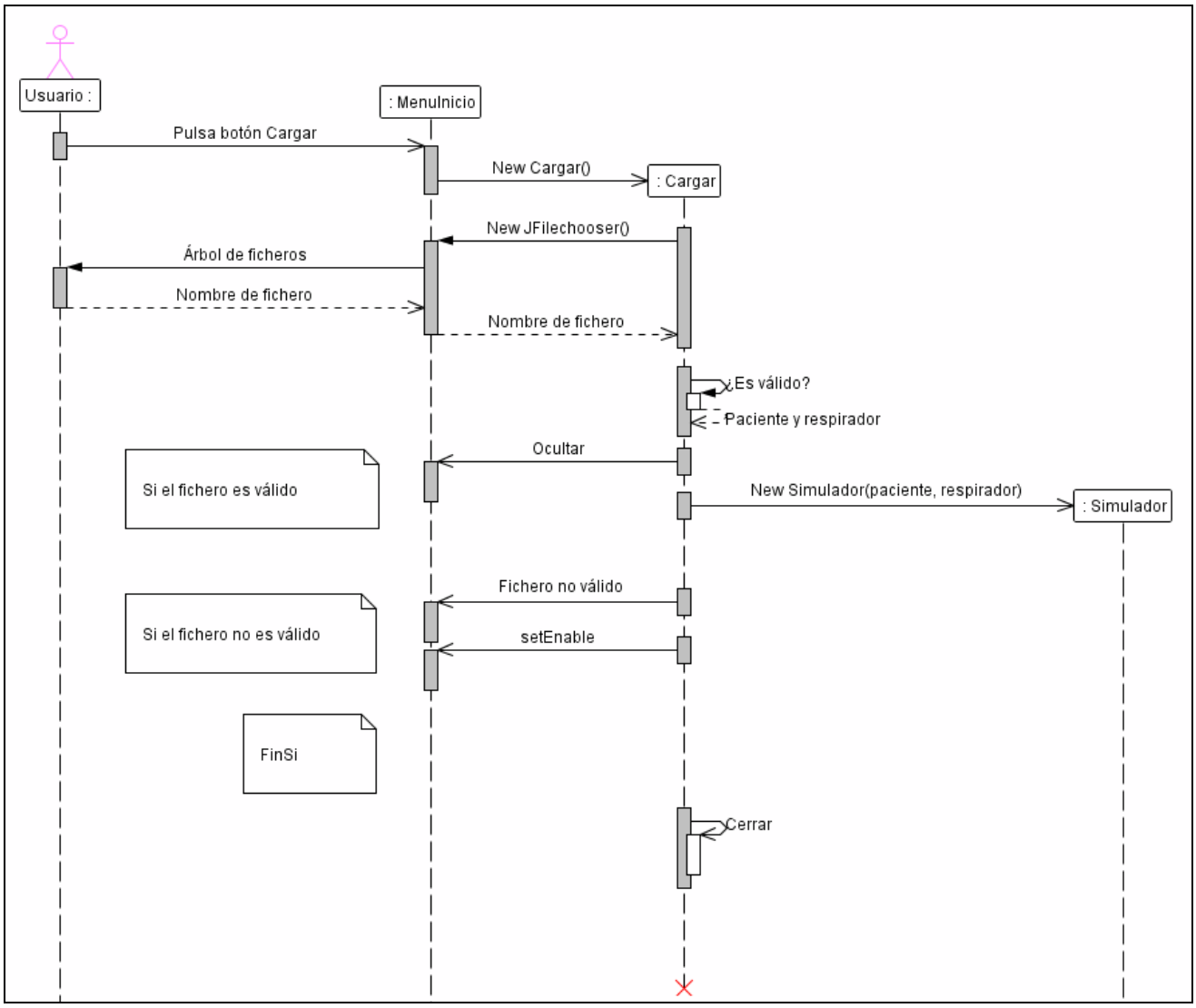


Figura 5.1.1 Diagrama de secuencia: Cargar simulación

En el diagrama anterior, se puede observar la secuencia de acontecimientos que se producen en el caso de uso Cargar simulación.

Al iniciarse la aplicación, el usuario tiene la posibilidad de cargar una simulación almacenada anteriormente. Para ello debe pulsar el botón “Cargar” el cual desencadena la secuencia de mensajes que devuelven al usuario el árbol de directorios para que éste pueda escoger el fichero que desea cargar.

Una vez elegido el fichero, el objeto Cargar comprueba que éste sea válido. Si lo es, se pasa el control de la ejecución al objeto Simulador enviando los objetos paciente y respirador que contiene el fichero, y oculta el menú de inicio. Si por el contrario, el fichero no es válido, el objeto Cargar envía un mensaje de error y pasa el control al menú de inicio. En ambos casos, el objeto Cargar es destruido al final de la ejecución.

El proceso de Cargar Simulación desde el menú de simulación no se ha representado debido a que es igual al anterior salvo que se inicia desde el objeto Simulador.

5.2. Diagrama de secuencia: Configuración de paciente y respirador

La figura 5.2 muestra el diagrama de secuencia Configuración de paciente y respirador.

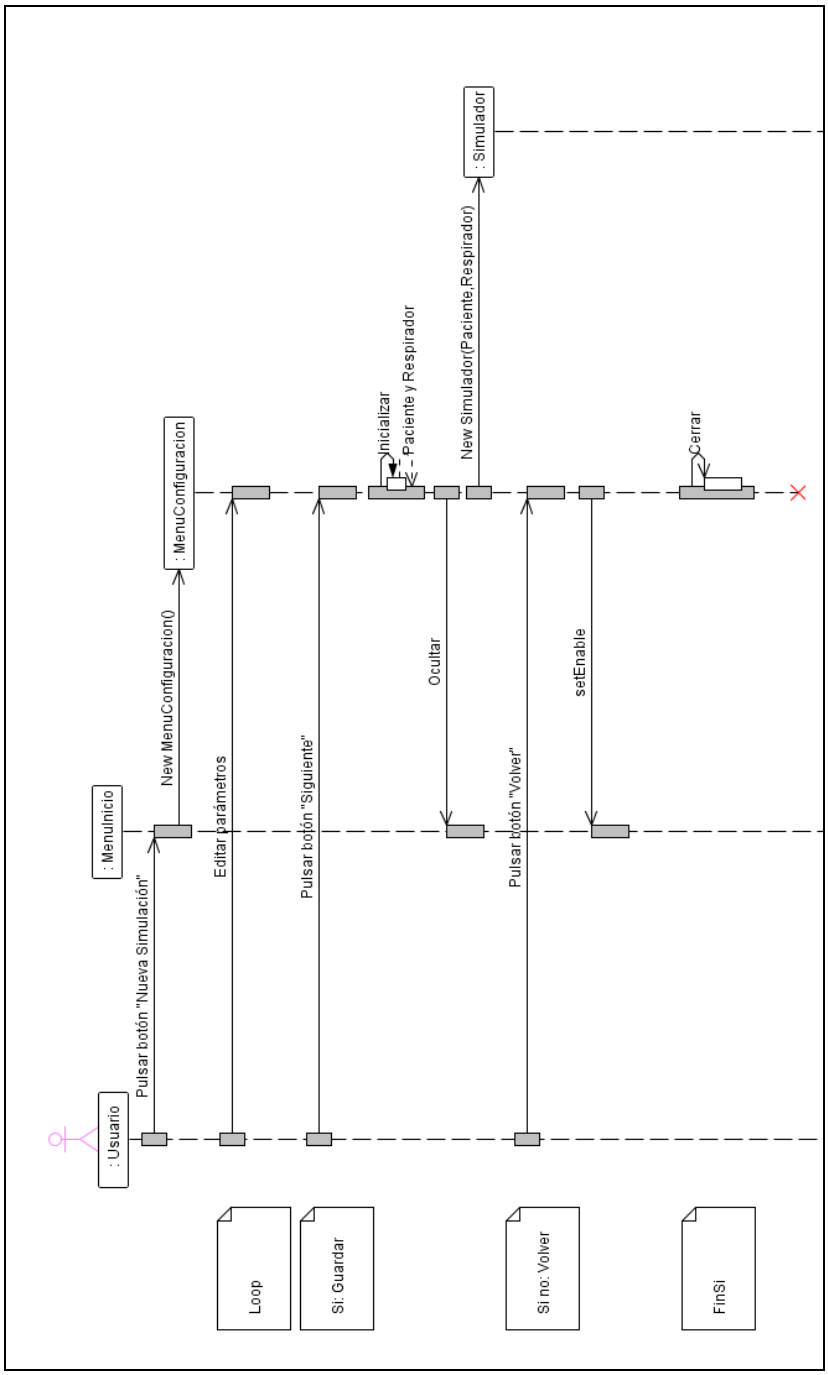


Figura 5.2.1 Diagrama de secuencia: Configuración de paciente y respirador

En el diagrama anterior se representa la secuencia de acontecimientos que se producen en el caso de uso Configuración de paciente y respirador.

Al iniciarse la aplicación, el usuario además de tener la posibilidad de cargar una simulación, como se ha comentado en el caso anterior, también puede crear una nueva partiendo de los valores por defecto.

La ejecución comienza cuando el usuario pulsa el botón “Nueva simulación”, desencadenando la creación del Menú de configuración por parte del Menú de inicio. Seguidamente el usuario puede modificar los diferentes parámetros tantas veces como quiera hasta que pulsa el botón “Siguiente”.

Cuando MenúConfiguración recibe el evento del botón, se envía un mensaje reflexivo creando los objetos paciente y respirador con los parámetros establecidos. Una vez creados, MenúConfiguración envía un mensaje a MenúInicio para que éste se oculte y seguidamente crea el objeto simulador pasándole los objetos paciente y respirador, y le cede el control.

Si en lugar de pulsar “Siguiente”, el usuario pulsa “Atrás”, se devuelve el control al MenúInicio.

En ambos casos, al finalizar la ejecución, el objeto MenúConfiguración es destruido.

El proceso de Cargar Simulación desde el menú de simulación no se ha representado debido a que es igual al anterior salvo que se inicia desde el objeto Simulador.

5.3. Diagrama de secuencia: Guardar

La figura 5.3 muestra el diagrama de secuencia Configuración de paciente y respirador.

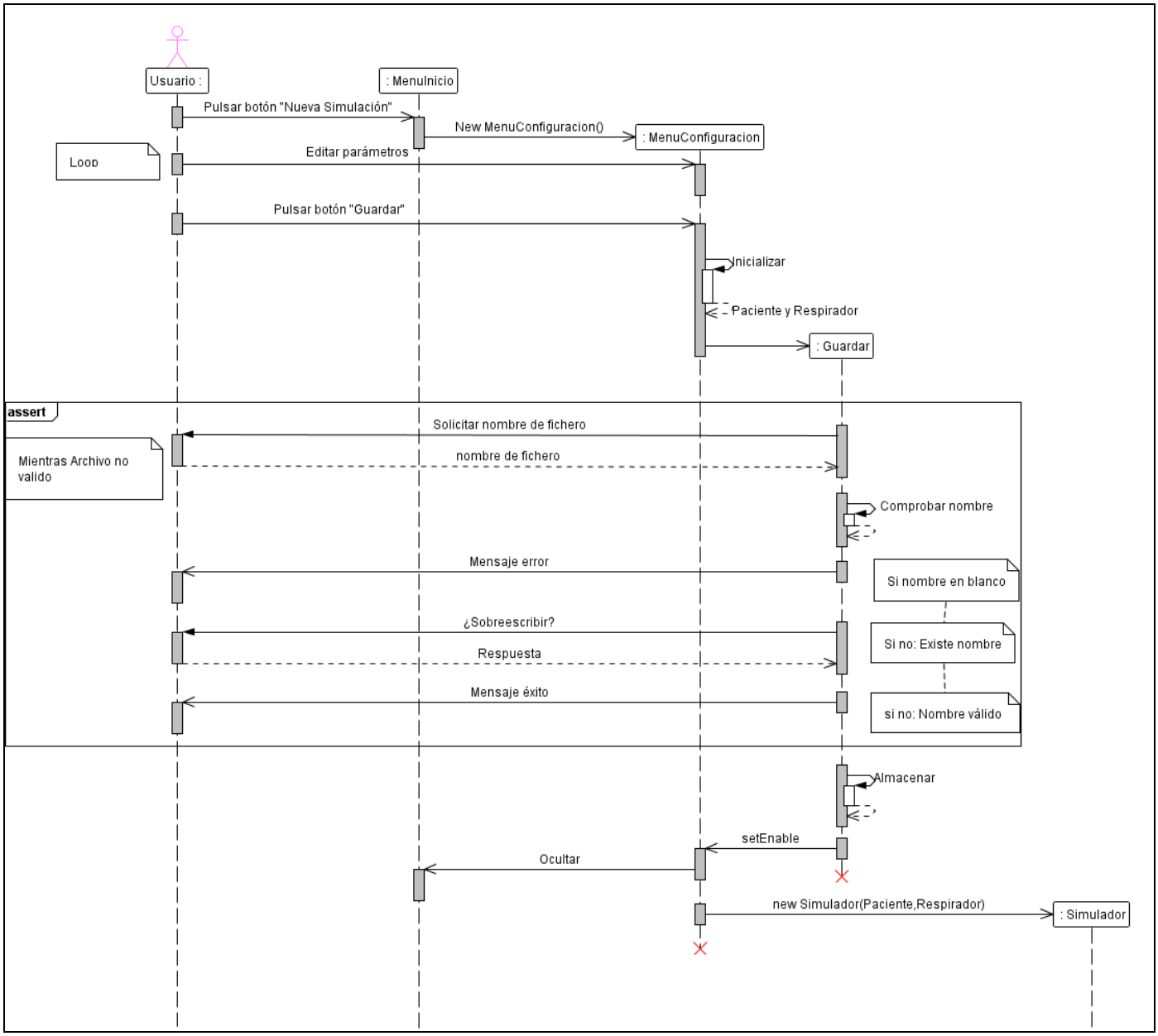


Figura 5.3.1 Diagrama de secuencia: Guardar

En el diagrama anterior, se puede observar la secuencia de acontecimientos que se producen en el caso de uso Guardar.

La ejecución comienza al igual que en el diagrama 5.2, con la acción pulsar botón “Nueva Simulación” por parte del usuario y modificando los valores de los parámetros. La diferencia radica en que la ejecución se realiza cuando el usuario pulsa el botón “Guardar” en el menú de configuración en lugar de “Siguiente”.

Una vez se ha enviado el evento del botón, el objeto MenúConfiguración se envía un mensaje reflexivo creando los objetos paciente y respirador con los parámetros establecidos.

Posteriormente crea el objeto Guardar el cual solicita al usuario que introduzca un nombre para el nuevo fichero. Una vez enviado el nombre, el objeto comprueba que sea un nombre válido. Si el nombre es una cadena vacía el objeto envía un mensaje de error y se queda a la espera de un nuevo nombre. Si existe un fichero con el mismo nombre pregunta al usuario si desea sobrescribirlo y se detiene la ejecución hasta obtener una respuesta. Por último, si el nombre es válido (o se desea sobrescribir), se almacena el fichero, se devuelve el control a MenúConfiguración y se destruye el objeto Guardar.

Para terminar, MenúConfiguración envía un mensaje a MenúInicio para que se oculte y seguidamente crea el objeto simulador pasándole los objetos paciente y respirador, y le cede el control.

El proceso de Guardar desde el menú de simulación no se ha representado debido a que es igual al anterior salvo que se inicia desde el objeto Simulador.

5.4. Diagrama de secuencia: Ejecutar simulación

A continuación se muestra la figura 5.4.1 donde se puede observar de forma simplificada la secuencia Ejecutar Simulación.

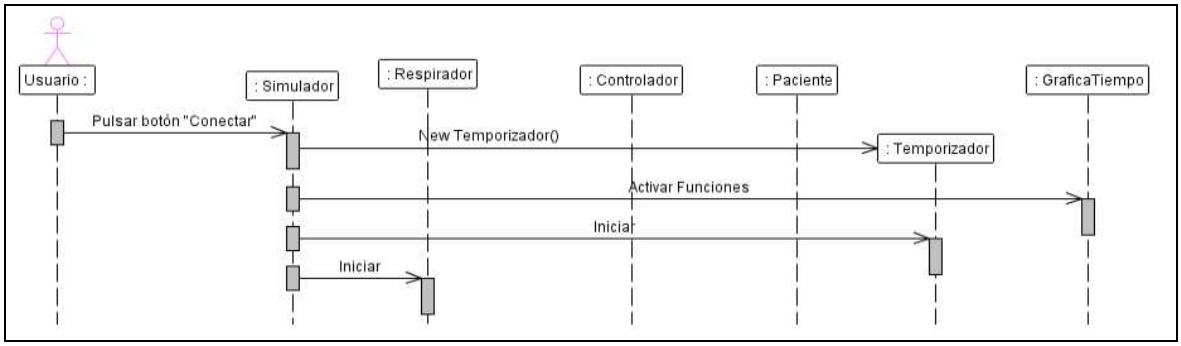


Figura 5.4.1 Diagrama de secuencia: Ejecutar simulación

Como se puede apreciar, la ejecución de la secuencia del diagrama comienza cuando el usuario pulsa el botón “Conectar”. En ese momento el Simulador lanza todos los eventos necesarios para comenzar con la simulación. En primer lugar crea el objeto Temporizador el cual se encarga del refresco de las gráficas y activa las funcionalidades de las gráficas. Posteriormente lanza los mensajes de inicio al temporizador y al respirador los cuales comienzan la ejecución de sus respectivos hilos (los dos son Timers) ejecutándose cada cierto tiempo determinado.

El diagrama anterior al basarse en el diagrama de casos de uso Ejecutar simulación, no detalla el proceso de simulación, solamente el comienzo de su ejecución. En la figura 5.4.2 se puede observar el diagrama anterior ampliado con los procesos que ocurren entre los diferentes objetos una vez comenzada la simulación. Este diagrama representa el “corazón” de la simulación, todo el flujo de mensajes necesario para poder realizar la simulación.

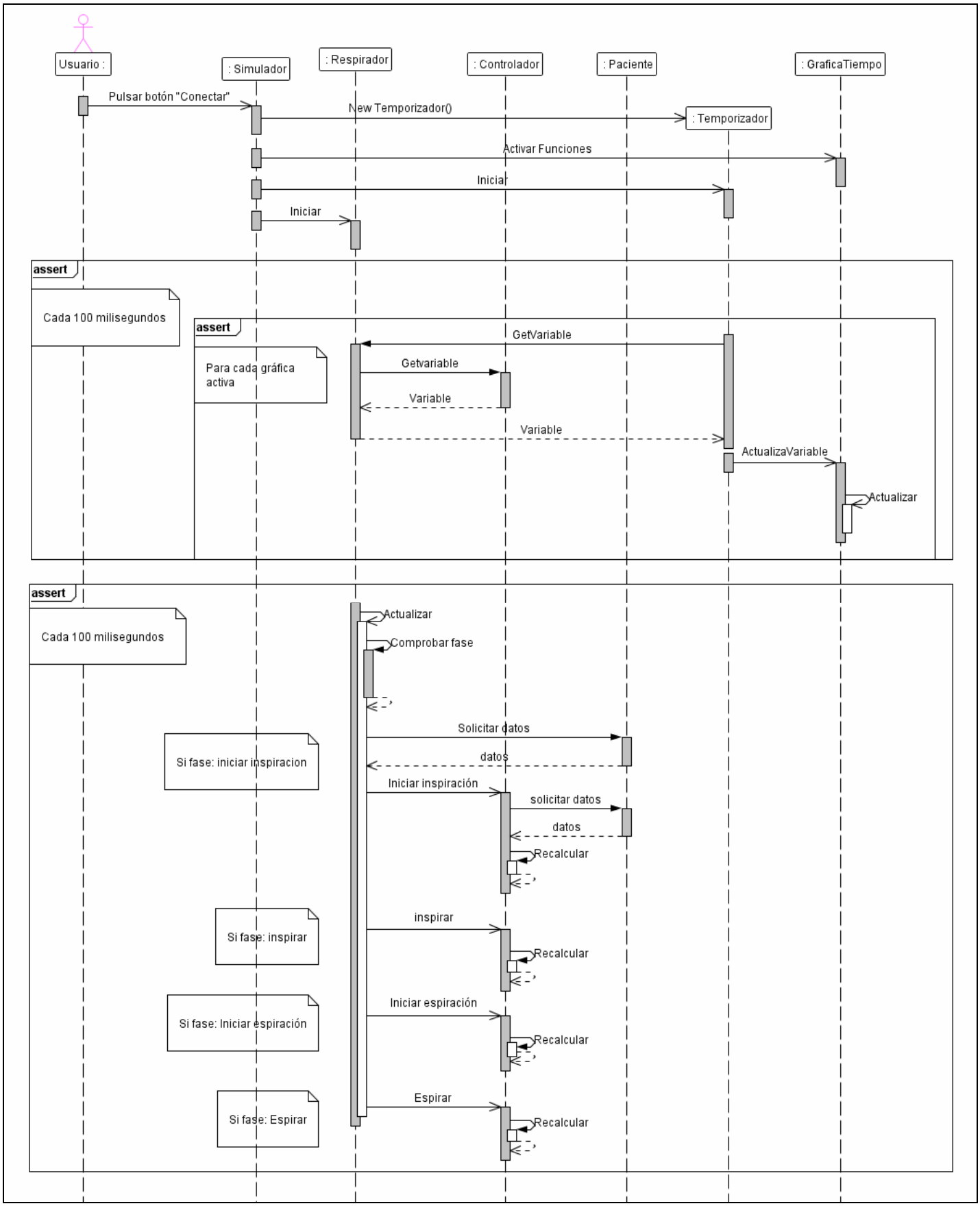


Figura 5.4.2 Diagrama de secuencia: Simulación

En el diagrama anterior se puede identificar claramente dos hilos de ejecución. El primero corresponde a la ejecución de la actualización y refresco de las gráficas, y el segundo a la simulación del proceso ventilatorio. Estos dos objetos son Timers que ejecutan el proceso cada 100 milisegundos de forma independiente.

En el proceso de actualización de las gráficas, el Temporizador se encarga de solicitar al Respirador los datos necesarios para actualizar cada gráfica. Una vez obtenidos los datos, envía un mensaje de actualización con el dato correspondiente a la gráfica. Al recibir los datos, la gráfica se envía un mensaje reflexivo añadiendo el nuevo dato al grafo. Este bucle se realiza hasta que se hayan actualizado todas y cada una de las gráficas activas.

En el proceso de simulación es el Respirador el encargado de controlar el tiempo transcurrido entre cada ejecución. Cuando comienza la ejecución del hilo, el Respirador se envía un mensaje reflexivo para actualizar su estado (tiempo) y comprobar en qué fase del ciclo respiratorio se encuentra la simulación. En este momento pueden ocurrir cuatro acciones distintas según la fase.

Cuando el estado es inicio de la inspiración, el Respirador solicita los datos estáticos del paciente por si han sido actualizados durante el transcurso del ciclo respiratorio y solicita al controlador la actualización de la simulación. El controlador a su vez solicita los datos del paciente que le son necesarios y realiza el cálculo de los parámetros variables de la simulación (presión ventilatoria, volumen corriente y flujo respiratorio) para ese instante de tiempo.

En los otros tres casos: inspiración, inicio de la espiración y espiración, el proceso de ejecución el mismo salvo por el mensaje enviado al Controlador en cada caso. Una vez el Controlador recibe el mensaje, recalcula los parámetros variables de la simulación para la fase correspondiente en ese instante de tiempo.

El proceso de funcionamiento interno de los objetos descritos anteriormente se explica en el siguiente capítulo.

5.5. Diagrama de secuencia: Editar gráficas

Como vimos en el capítulo anterior, la descripción del caso de uso Editar gráficas está compuesto por otros cuatro casos de uso. Se ha decidido representar únicamente el diagrama de Mostrar gráficas ya que es el único que tiene cierta complejidad puesto que los demás no realizan envíos de mensajes entre las clases principales si no que toda la secuencia la ejecuta la clase GráficaTiempo o sus derivadas.

A continuación se muestra la figura 5.5 que corresponde con el diagrama de secuencia Mostrar gráfica.

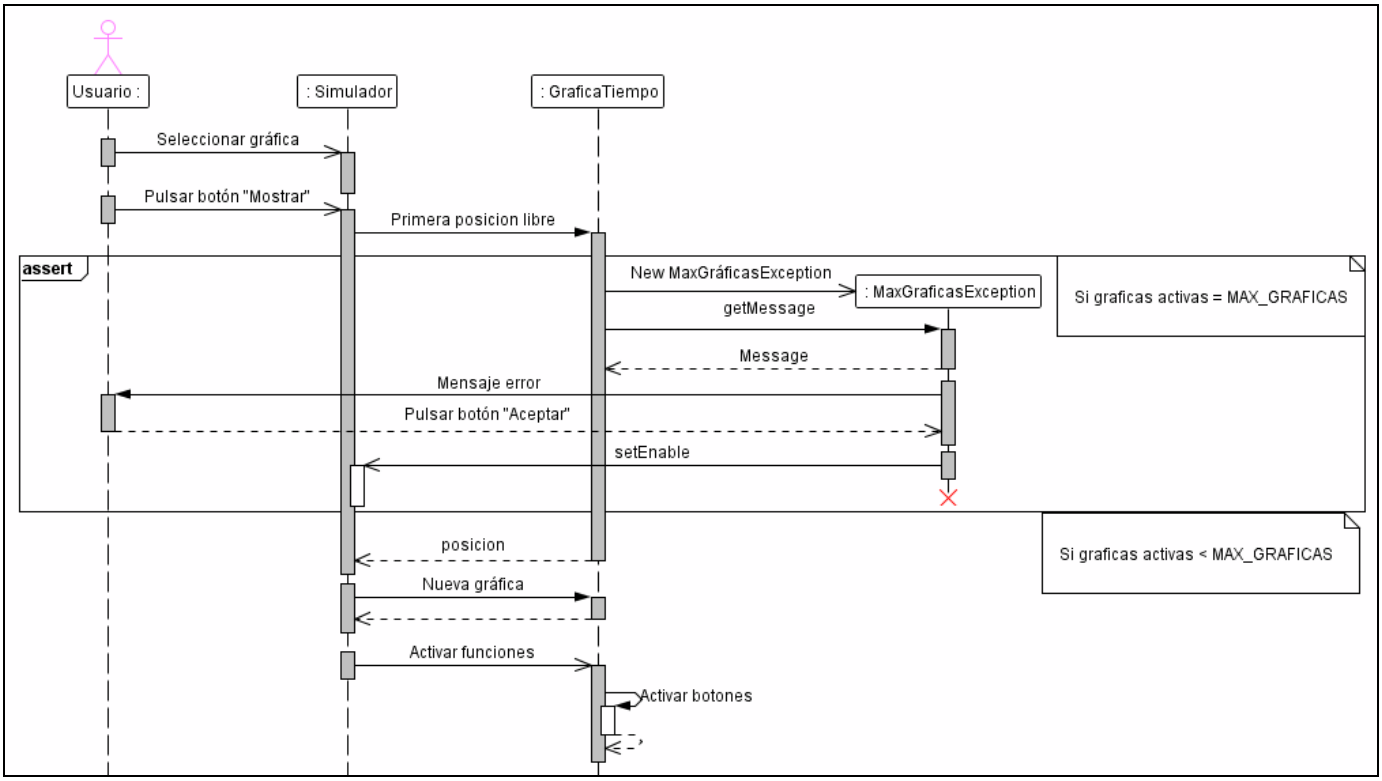


Figura 5.5.1 Diagrama de secuencia: Mostrar gráfica

Como se puede observar, la ejecución comienza cuando el usuario selecciona una gráfica de la lista desplegable y pulsa el botón “Mostrar”. En ese momento el Simulador envía un mensaje a GráficaTiempo solicitando que le devuelva la primera posición libre para instanciar una gráfica. En este punto pueden ocurrir dos ejecuciones diferentes.

Si la acción “Primera posición libre” no encuentra una posición libre lanza una excepción de la clase MaxGráficasException y solicita el mensaje de la excepción. La excepción a su vez lanza un mensaje de error al usuario y espera la confirmación de éste hasta que pulse el botón “Aceptar”. Por último, la excepción muere y se devuelve el control al Simulador.

Por el contrario, si existe una posición libre para una nueva gráfica, se devuelve el número de la posición al Simulador y éste crea una nueva gráfica y solicita activar todas sus funciones. La nueva gráfica al recibir el mensaje activa todos los botones con las respectivas funciones que tienen disponibles.

Capítulo 6. DISEÑO

El objeto de este capítulo es describir el diseño de la aplicación partiendo del análisis desarrollado en el capítulo anterior.

En primer lugar se expone el diseño de la interfaz gráfica de usuario y seguidamente un análisis completo de las clases más relevantes y los algoritmos de simulación.

6.1. Diseño de la interfaz gráfica de usuario

La interfaz gráfica de la aplicación se basa en una serie de formularios (menús) donde se proporciona el acceso a todas las funcionalidades descritas en los casos de uso del capítulo 3.

Para el desarrollo de la interfaz se ha utilizado la librería gráfica SWING de Java utilizando varios de los objetos que dispone como por ejemplo JFrame, JInternalFrame, JPanel, JButton, JSpinner…

Además de la librería propia de Java, tal y como se muestra más adelante, las gráficas usadas en el menú de simulación han sido realizadas con la librería open source JFreeChart.

6.1.1. Menú de inicio

Al iniciar la aplicación el primer formulario que aparece es el menú de inicio. Desde este menú el usuario puede acceder a las funciones de crear una nueva simulación y a la de cargar una simulación almacenada previamente. Además tiene un botón en la parte inferior que le permite cerrar la aplicación. Esta primera ventana ha sido diseñada con un tamaño óptimo para cualquier resolución de pantalla, incluso las más bajas (800x600), no es redimensionable y no posee barra de titulo adquiriendo así un matiz más parecido a una splash screen que a una ventana propiamente dicha, consiguiendo un diseño más atractivo.

Figura 6.1.1 Captura: Menú de inicio

6.1.2. Cargar simulación

El menú de cargar simulación es bastante simple. Únicamente se utiliza un objeto del API de java, el JFileChooser, donde permite navegar por el árbol de directorios del sistema operativo para escoger el fichero que se desea cargar. Por defecto, se muestra el directorio "pacientes" de la aplicación, lugar donde se almacenan los ficheros de simulación. Además, para facilitar la búsqueda, implementa un filtro para ver únicamente los ficheros .pvm de la aplicación.

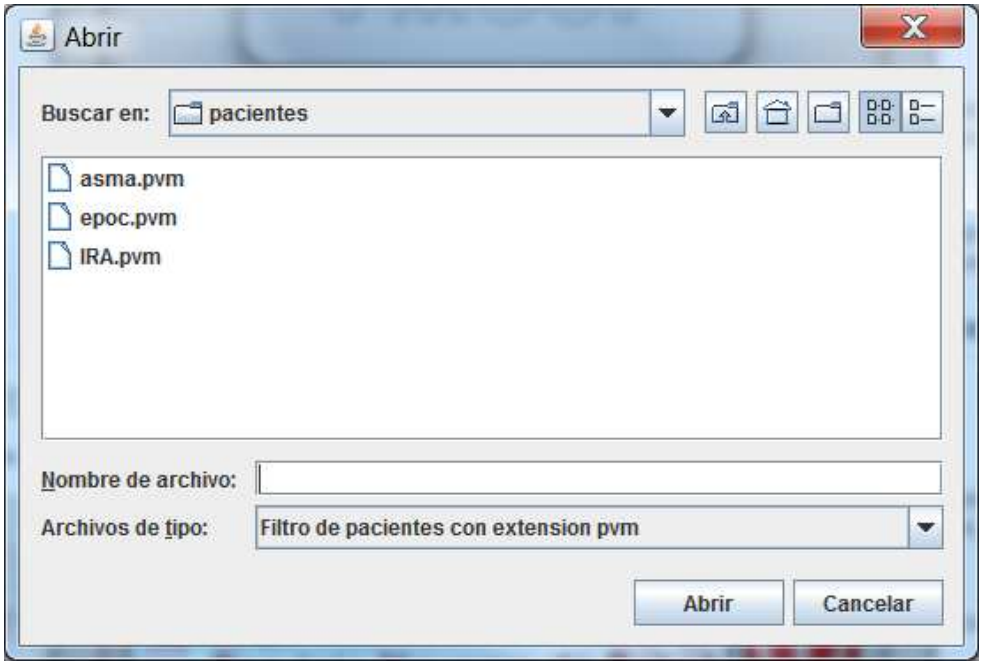


Figura 6.1.2 Captura: cargar simulación

6.1.3. Menú de configuración

El menú de configuración de la simulación se divide a grandes rasgos en tres partes. Por un lado está el JTabbedPane que contiene los dos paneles de configuración de la simulación, y por el otro, en la parte inferior de la ventana se encuentran los botones de navegación.

Figura 6.1.3 Captura: Configurar paciente

En la primera pestaña del JTabbedPane se encuentran los parámetros del paciente que se pueden configurar. Estos se encuentran divididos en dos grupos, los parámetros que influyen en la simulación de la ventilación mecánica, y los parámetros que pertenecen a la simulación de la máquina de gases.

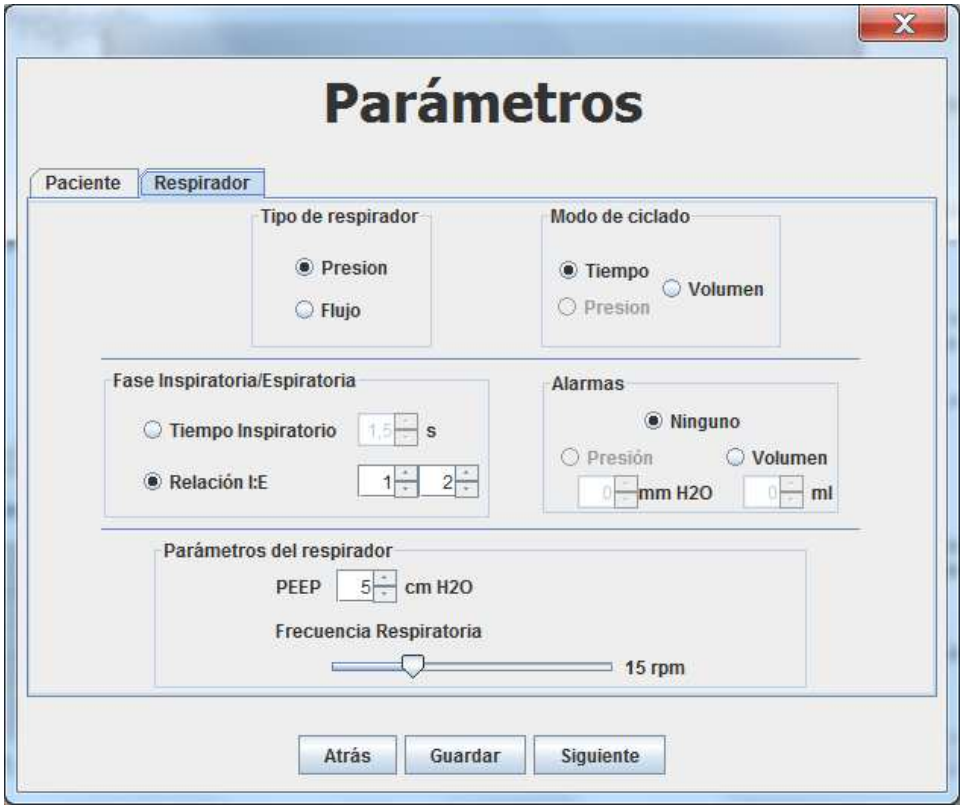


Figura 6.1.4 Captura: Configurar respirador

En la segunda pestaña del JTabbedPane se puede observar más subgrupos que en la anterior. Aquí se encuentran los parámetros del respirador. El panel se divide en tres líneas, las dos primeras permiten elegir la modalidad del respirador mientras que la última son los parámetros regulables durante la simulación. Como se aprecia en la imagen y durante el uso de la aplicación, no todas las opciones de configuración están disponibles para todas las modalidades de funcionamiento. Por ejemplo, si está activo el tipo presión, el modo de ciclado por presión esta deshabilitado debido a que son incompatibles.

Por último, los botones de navegación de la parte inferior, permiten volver al menú de inicio, guardar la configuración de la simulación y crear los objetos de simulación. Al terminar las dos últimas operaciones, se abre el menú de simulación.

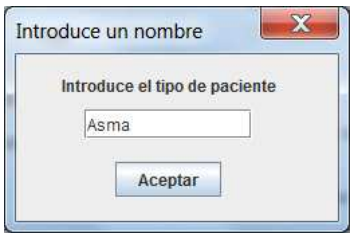


Figura 6.1.5 Captura: Guardar

6.1.4. Menú de simulación

El menú de simulación es donde se muestran los datos obtenidos durante el proceso de simulación y contiene todas las funciones principales de interacción del usuario con la aplicación.

Figura 6.1.6 Captura: Menú de simulación

Como se puede observar en la Figura 6.1.5, existe una división de la ventana en dos partes claramente diferenciadas.

En la parte izquierda se encuentra el panel de configuración. En el centro del panel, el usuario puede cambiar el valor de los parámetros de la simulación, los cuales se encuentran ordenados en diferentes pestañas según su naturaleza. En la parte superior se encuentra la lista desplegable con las diferentes gráficas disponibles junto con el botón para mostrarlas y la variable de control del respirador (presión o volumen según el tipo de respirador elegido). Y en la parte inferior, se encuentran los botones para volver al menú de inicio, salir de la aplicación y comenzar la simulación.

La parte derecha de la ventana contiene el panel de las gráficas. Este panel puede contener hasta 3 gráficas simultáneamente, cada una de ellas con sus respectivos botones para activar o desactivar las diferentes funcionalidades que poseen.

Figura 6.1.8 Captura: Gráfica maximizada

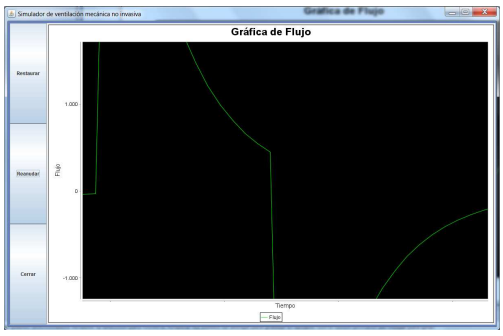


Figura 6.1.7 Captura: Zoom gráfica

Si se desea cerrar una gráfica, ésta dejará libre el espacio que ocupaba permitiendo abrir una nueva gráfica.

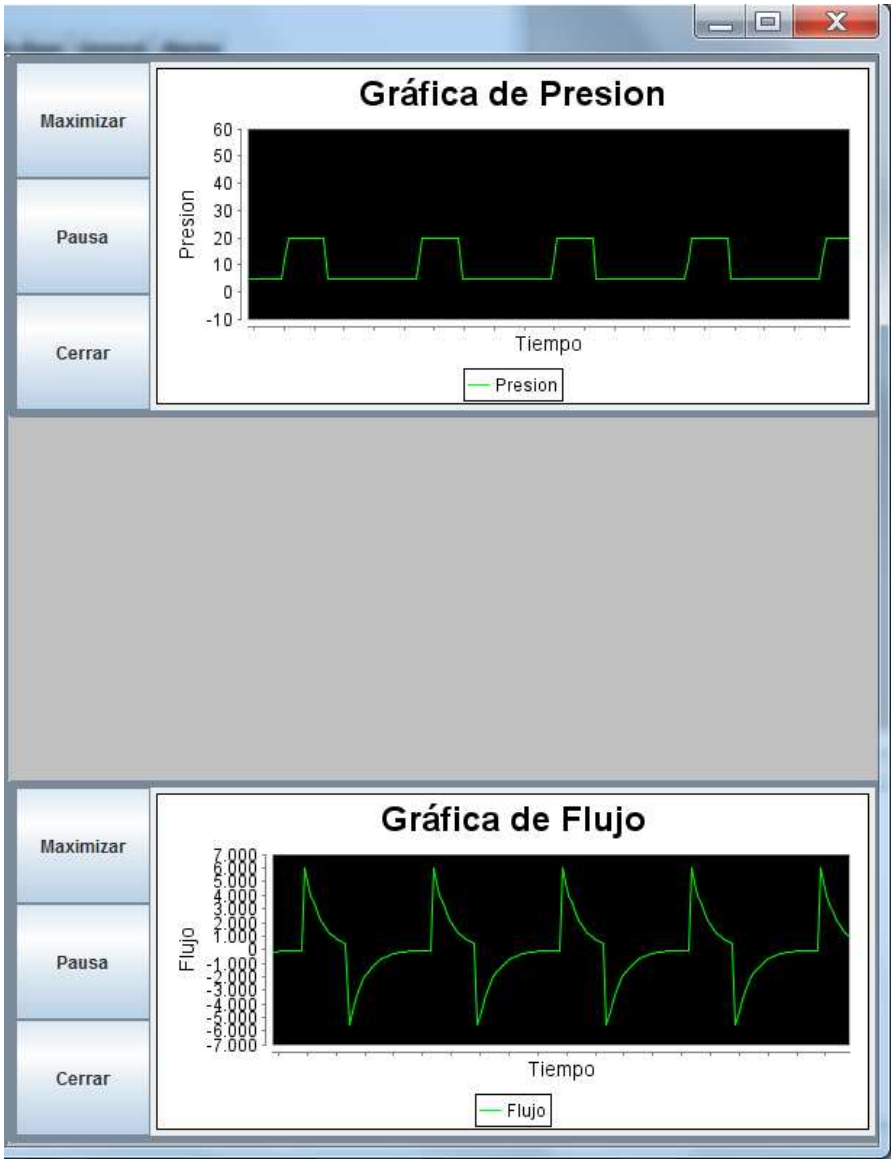


Figura 6.1.9 Captura: Panel de gráficas

Una de las características a reseñar de este apartado es la capacidad de la interfaz para adaptarse a los diferentes tamaños y resoluciones. Si se compara la Figura 6.1.5 con la Figura 6.1.10, se puede observar la misma ventana en diferente resolución o tamaño. En la Figura 6.1.10, al redimensionar la ventana completa, las gráficas son redimensionadas para ocupar el espacio proporcional al nuevo tamaño sin perder en ningún momento la visibilidad de las tres gráficas. Lo mismo ocurre cuando una de las gráficas está maximizada. Esta funcionalidad de la interfaz ha sido realizada para cumplir el requisito de la adaptación a los diferentes modos y resoluciones de pantalla.

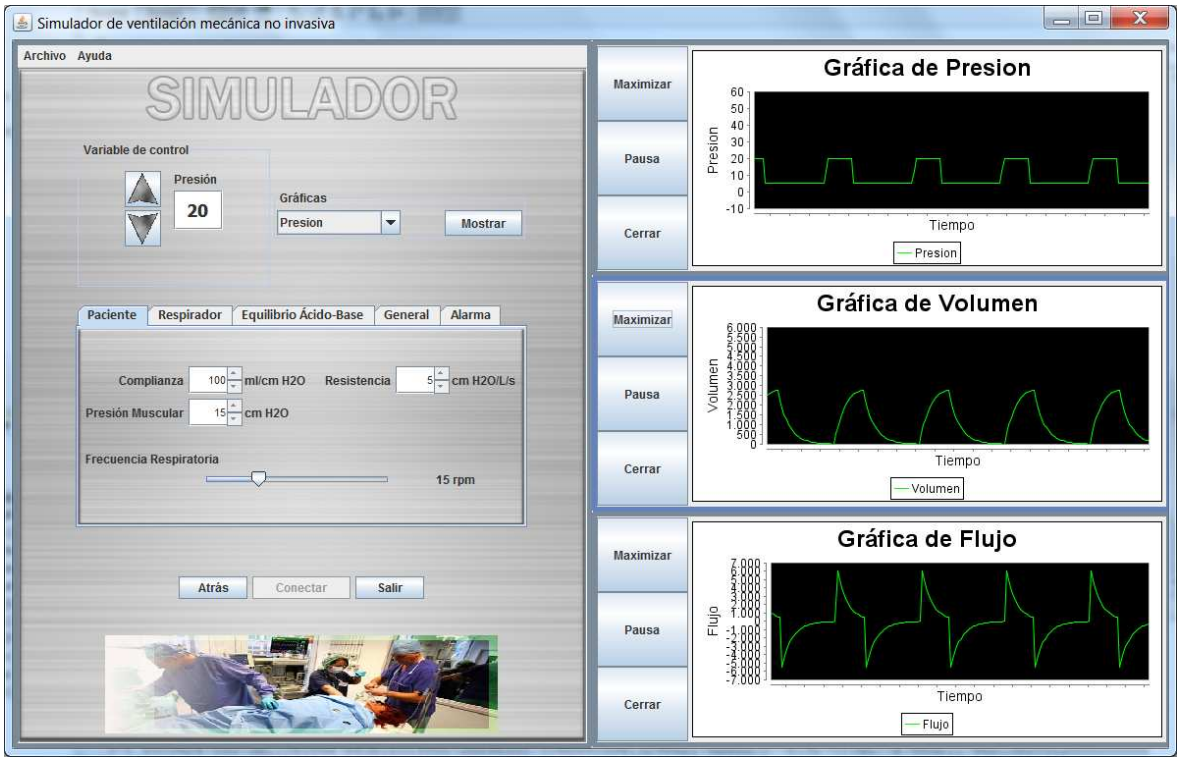


Figura 6.1.10 Captura: Menú de simulación

6.2. Diseño y arquitectura de las clases

La aplicación, cuando menos las clases de simulación, se basa en una arquitectura modular. Se ha realizado el diseño de la aplicación de forma que ésta pueda ser ampliada fácilmente con un mayor número de opciones de configuración. Es decir, permite a través de interfaces y clases abstractas una rápida implementación de nuevos tipos de control (actualmente están implementados presión y volumen), de ciclado, de alarmas… Además, el diseño modular no solo afecta al conjunto de clases de simulación, si no que el paquete de las gráficas también ha sido diseñado de forma modular dando la posibilidad de ampliar ciertos aspectos como el número de variables a representar en un solo gráfico (actualmente sólo son necesarias hasta dos series en un gráfico).

La arquitectura de la aplicación se divide en cuatro paquetes según las funcionalidades de las clases que contienen.

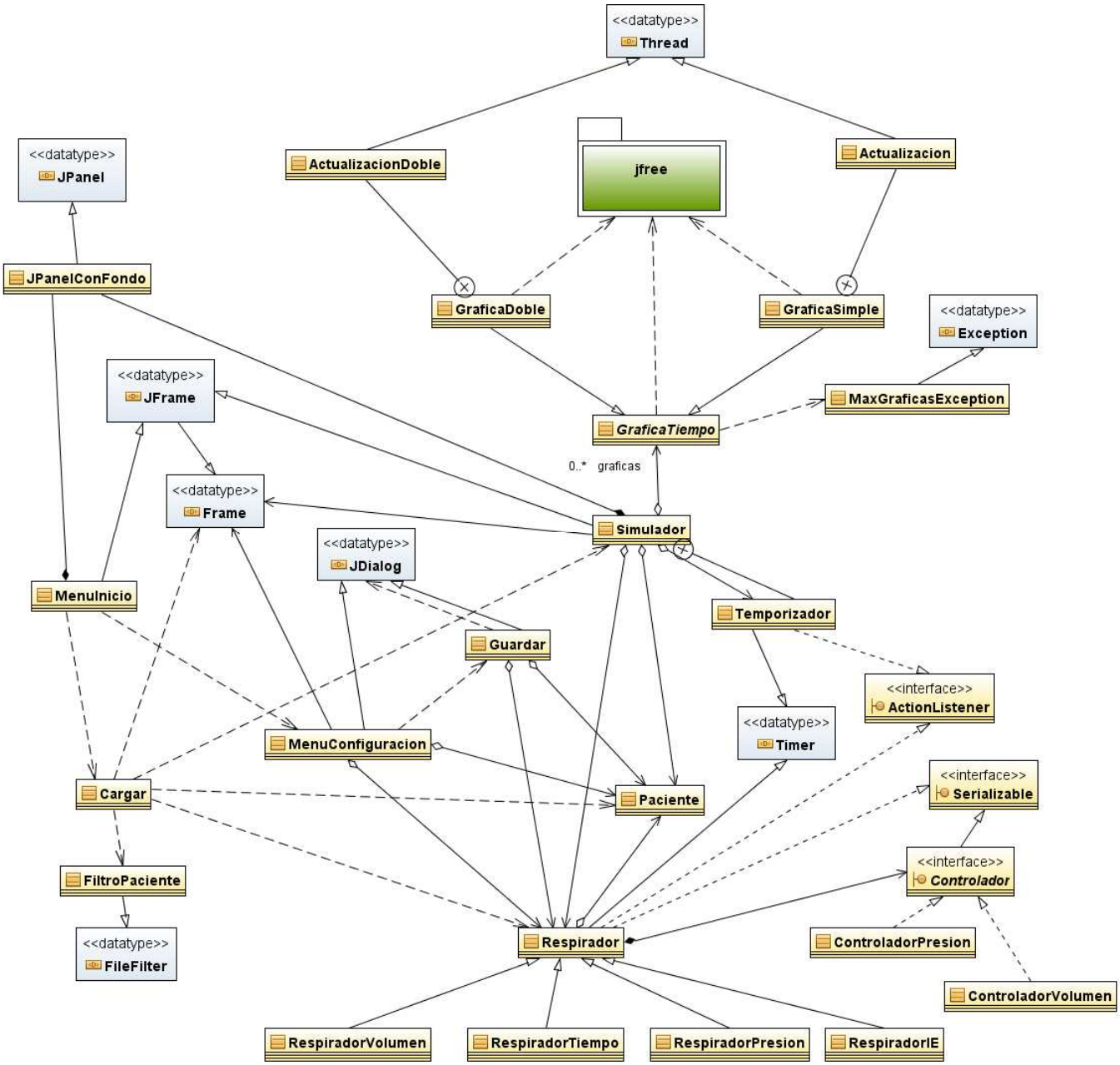
En el paquete formularios se encuentran todas las clases que heredan o instancian componentes form de SWING además de las clases que utilizan de apoyo.

El segundo paquete contiene las imágenes que utiliza la aplicación y una clase que hereda de JPanel y que permite establecer imágenes de background en la aplicación.

El paquete gráficas contiene las clases necesarias para construir las gráficas que representarán los valores de la simulación.

Por último, el paquete respirador es el paquete principal de la simulación. En él se contienen las tres clases (o interfaces) principales y las subclases que implementan los algoritmos de simulación.

En el siguiente diagrama de clases se puede observar de forma global la organización de la arquitectura.



En un primer momento puede resultar un poco confusa al haber tantas relaciones entre clases, pero si se observa detenidamente se puede comprobar que en la parte superior derecha se encuentran las clases del paquete gráficas, en la parte inferior derecha todas las clases correspondientes a la simulación y el resto son las clases correspondientes a todos los formularios y los componentes de estos.

A continuación se realiza un análisis más detallado de la arquitectura.

6.2.1. El Package Formularios

Como se ha descrito anteriormente, en este paquete se encuentran todas las clases correspondientes a los formularios de la aplicación. Estas clases son:

- **MenúInicio:** Clase que hereda de JFrame y construye el menú de inicio de la aplicación, el cual contiene las opciones de crear un nuevo paciente, cargar un paciente o salir de la aplicación

- **Cargar:** Esta clase se encarga de instanciar un JFileChooser para elegir un fichero del disco, comprueba que el fichero leído es correcto y si es así carga su contenido en un objeto paciente y un objeto respirador. Si el proceso ha funcionado correctamente se cierra el form anterior (padre) y se instancia la clase Simulador. Si ha ocurrido algún error durante el proceso, ya sea por fallo de lectura, datos corruptos... la clase devuelve el control al form padre que lo ha invocado. La clase está diseñada con una metodología estructurada más que orientada a objetos ya que todas las acciones se llevan a cabo en el constructor y la clase no posee ningún otro método.

- **FiltroPaciente:** Clase que filtra por extensión los ficheros leídos que recibe. Normalmente la clase se pasa como argumento a un JFileChooser.

- **MenúConfiguración:** Esta clase hereda de JDialog y crea el formulario de configuración del paciente y el respirador. La clase tiene varios paneles con los distintos parámetros del paciente y del respirador que se permiten configurar. Contiene tres botones para cada una de las opciones disponibles: volver al menú de inicio, abrir el menú de guardar la configuración o iniciar el menú de simulación.

- **Guardar:** Clase que hereda de JDialog y crea el formulario donde se ingresa el nombre para el archivo de configuración del paciente a guardar. Si el nombre es válido, guarda los objetos paciente y respirador mediante serialización.

- **MenúInterno:** Clase que hereda de JInternalFrame y modifica el aspecto de éste eliminando la barra de título. La clase es usada como una plantilla.

- **Simulador:** La clase Simulador es la clase que contiene todos los elementos que representan la simulación (paciente, respirador, paneles de configuración, gráficas y opciones de menú). La clase recibe como parámetro el objeto paciente y el objeto respirador instanciados en la clase padre y prepara el formulario para iniciar la simulación. Cuando se activa la simulación, se inicia el respirador y se realiza una instancia de la clase interna Temporizador la cual refresca las gráficas que muestran los valores del paciente. Esta clase permite instanciar las gráficas, configurar los parámetros del objeto paciente y del objeto respirador, volver al menú de inicio, cargar y guardar la configuración de un paciente/respirador, crear una nueva simulación y salir de la aplicación.

- **Temporizador.** Clase interna de Simulador que hereda de Timer e implementa la interfaz ActionListener. Controla el tiempo de refresco de las gráficas y con qué valores deben actualizarse.

No es necesario explicar todos y cada uno de los atributos y métodos que contienen estas clases debido a que muchos de ellos son eventos, de los diferentes componentes (paneles, botones, spinner…) que despliegan un nuevo formulario (visto en el diseño de la interfaz) o que actualizan los datos de una determinada clase (ej: cambio de valor del atributo resistencia de la clase paciente).

Una de las partes que pueden resultar más confusas son las relaciones de dependencia o asociación entre los distintos frames o jdialogs. Estas relaciones son debidas a que al instanciar una nueva clase JFrame o JDialog, ésta recibe como parámetro el frame que la ha instanciado. Esta acción se realiza para que el formulario activo, al terminar su ejecución, pueda devolver el control al objeto que lo invocó o al objeto principal (en este caso el MenúInicio).

Uno de los puntos más importantes ocurre en las clases Cargar y MenúConfiguración. Estas clases son las encargadas de crear por primera vez los objetos Paciente y Respirador los cuales componen la base de la simulación, además de crear la instancia del objeto Simulador.

En el caso del objeto Cargar, éste lee el contenido del fichero elegido por el usuario y mediante el método readObject() del objeto ObjectInputStream recupera los objetos almacenados en él.

En el caso del objeto MenúConfiguración, cuando el usuario lanza el evento del botón “Guardar” o “Siguiente”, se lanza el método inicializar(), el cual instancia los objetos Paciente y Respirador con los valores establecidos por el usuario en los diferentes componentes del objeto (JSpinner, JSlider, JRadioButton…).

Por último, la clase más importante (y más utilizada), es la clase Simulador. Como ya se ha explicado, es la clase que actúa de interfaz entre los 3 grandes componentes de la aplicación: los formularios, las gráficas y la simulación.

6.2.2. El Package Imágenes

Este paquete es el más simple de los cuatro. Apenas contiene código fuente, solamente una clase que es utilizada por los objetos MenúInicio y Simulador:

- **JPanelConFondo:** Clase que hereda de JPanel y permite cargar una imagen como fondo.

El resto de los objetos del paquete son imágenes utilizadas por el JPanelConFondo, JButtons y JLabels.

6.2.3. El Package Gráficas

En este paquete se encuentran las clases que permiten representar mediante gráficas en tiempo real los valores producidos por la simulación.

Uno de los puntos clave de este paquete es la modularidad. Las gráficas han sido diseñadas de tal forma que no dependen del objeto a representar por lo que pueden ser reutilizadas en otras aplicaciones siempre y cuando el formato del formulario sea igual al

objeto Simulador (gráficas en la parte derecha de la ventana y máximo tres gráficas simultáneas). De todas formas, si se desea poder instanciar más gráficas o que éstas sean independientes unas de otras, los cambios en el código serían mínimos.

El siguiente diagrama de clases muestra la estructura del paquete y los atributos y métodos de cada clase:

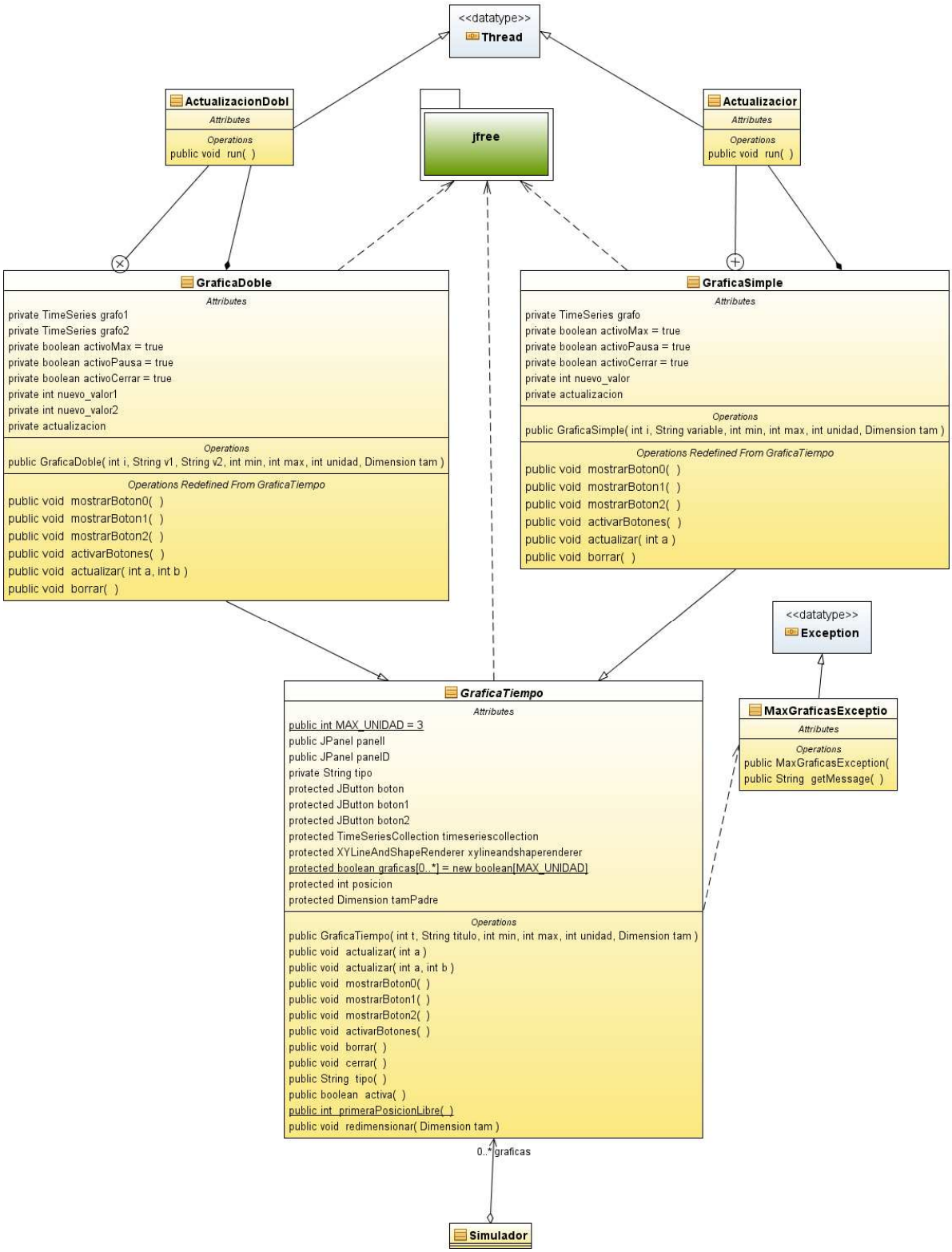


Figura 6.2.1 Diagrama de clases: Gráficas

- **GráficaTiempo:** Clase abstracta que actúa de contenedor de la gráfica y que implementa los métodos comunes y define los que deben ser implementados en las subclases. La clase hereda de JInternalFrame el cual es el marco donde se construye la gráfica. Esta clase hace uso de las librerías JFreeChart para construir los diferentes componentes que constituyen la gráfica.

- **GráficaSimple:** Clase que hereda de GráficaTiempo. Genera una serie de tiempo y la asocia a una gráfica y cada nuevo valor del eje Y se asocia al valor de tiempo correspondiente en milisegundos.

- **GráficaDoble:** Clase que hereda de GráficaTiempo. Permite generar dos series de tiempo simultáneamente en una misma gráfica y cada nuevo valor del eje Y se asocia al valor de tiempo correspondiente en milisegundos.

- **MaxGráficasException:** Clase que hereda de Exception usada para el control del número de gráficas.

Como ya se ha mencionado en varias ocasiones, la clase GráficasTiempo, y por la tanto las subclases, hacen uso de una librería externa open source llamada JFreeChart. Esta librería permite construir diferentes tipos de gráficos estáticos y definir una gran cantidad de parámetros de visualización y aspecto.

La aplicación en concreto, utiliza uno de los gráficos incluidos en la librería para representar series de tiempo. En primer lugar este gráfico es utilizado para representar gráficas de tiempo estáticas, como por ejemplo gráficos de economía por año, pero utilizando la unidad de tiempo second y millisecond se ha conseguido crear un efecto de gráfico en tiempo real. Aunque se renderiza una nueva imagen cada vez que se actualiza el gráfico, no genera ningún problema ya que la carga necesaria para realizar la acción es mínima.

Para más detalles sobre la API de la librería consultar las páginas web que aparecen en la bibliografía al final del documento.

La clase GráficaTiempo es una clase abstracta que contiene la especificación de todos los métodos que deben implementarse en las clases que heredan de ella. Además, esta clase contiene las operaciones base para poder instanciar el gráfico y todas ellas se realizan en el constructor. Estas operaciones son: la creación de los objetos necesarios para la construcción del gráfico (JFreeChart, ChartPanel…), la personalización de los ejes, la elección del render (XYLineAndShapeRendere), apariencia de la ventana, número de botones…

Otro de los métodos más importantes que implementa la clase es el redimensionar(Dimension tam); este método recibe como argumento la dimensión del objeto padre, por lo que es invocado cuando el padre cambia de tamaño, y su función es redimensionar la gráfica en proporción al nuevo tamaño del padre.

Las subclases, en este caso GráficaSimple y GráficaDoble, son las que terminan de construir el gráfico definiendo qué y cuantas Series van a utilizar y el aspecto que van a tener. Además, implementan los métodos definidos en la superclase. Estos métodos están asociados a los eventos de los botones e implementan las funcionalidades que se desea que tengan las gráficas. En este caso, se implementan las funcionalidades de pausa, cerrar y maximizar en las dos clases.

GráficaSimple y GráficaDoble contienen cada una de ellas una clase interna que hereda de Thread. Estas clases sobrescriben el método Run() con una llamada al método

addOrUpdate() del objeto TimeSerie para actualizar la gráfica. De esta forma, a la hora de actualizar las tres gráficas mediante el objeto Temporizador, el proceso no se realiza de forma secuencial si no mediante hilos. La decisión de implementar la función de actualización mediante hilos se debe al mejor aprovechamiento de los nuevos procesadores con varios núcleos y soporte multithreading.

Por último, se implementa una nueva excepción MaxGráficasException, que es lanzada por el constructor de la clase GráficaTiempo y por el método primeraPosiciónLibre() cuando no hay espacio disponible para instanciar más gráficas.

6.2.4. El Package Respirador

En el paquete respirador se encuentran las clases que componen el núcleo de la simulación. Son las clases que implementan los métodos y algoritmos necesarios para realizar la simulación de los diferentes tipos y modos de funcionamiento de un ventilador mecánico descritos en los capítulos anteriores.

- **Paciente:** La clase paciente contiene toda la información referente a los distintos parámetros inherentes al paciente y que intervienen de algún modo en el cálculo de la simulación. La clase consiste mayoritariamente en un almacén de información con diversos atributos y sus correspondientes métodos get y set. En esta clase se calcula el equilibrio acido-base ya que las variables que lo determinan son propias del paciente y no se ven afectadas por cambios del respirador.

- **Respirador:** La clase abstracta Respirador hereda de la clase Timer e implementa las interfaces ActionListener y Serializable. El propósito de la clase es controlar el tiempo invocando el método actionPerformed() cada 100 milisegundos el cual controla en qué momento del ciclo respiratorio se encuentra el paciente (inicio de la inspiración, inspiración, inicio de la espiración o espiración). Controla también la alarma si ésta se encuentra activada y si el respirador debe funciona en modo asistido o controlado.

- **RespiradorIE:** La clase RespiradorIE hereda de la clase Respirador. El control de las fases está determinado por la relación I:E establecida.

- **RespiradorTiempo:** La clase RespiradorTiempo hereda de la clase Respirador. El control de las fases está determinado por el tiempo inspiratorio.

- **RespiradorVolumen:** La clase RespiradorVolumen hereda de la clase Respirador. El ciclado está controlado por el límite de volumen. Solo funciona con respiradores controlados por presión.

- **RespiradorPresión:** La clase RespiradorPresión hereda de la clase Respirador. El ciclado está controlado por el límite de presión. Solo funciona con respiradores controlados por volumen.

- **Controlador:** Interfaz que definen todos los métodos necesarios para controlar la variación de cada uno de los parámetros respiratorios en cada instante de tiempo. Las clases que implementen esta interfaz son las encargadas de realizar los cálculos

matemáticos para la obtención de los valores de la simulación. Esta interfaz hereda de la clase Serializable ya que forma parte de la clase Respirador la cual es serializada para su almacenamiento.

- **ControladorPresión:** Clase que hereda de la interfaz Controlador. La clase implementa los métodos para cada una de las fases respiratorias de un respirador en la modalidad de presión.

- **ControladorVolumen:** Clase que hereda de la interfaz Controlador. La clase implementa los métodos para cada una de las fases respiratorias de un respirador en la modalidad de volumen.

El siguiente diagrama de clases muestra la estructura del paquete y los atributos y métodos de cada clase:

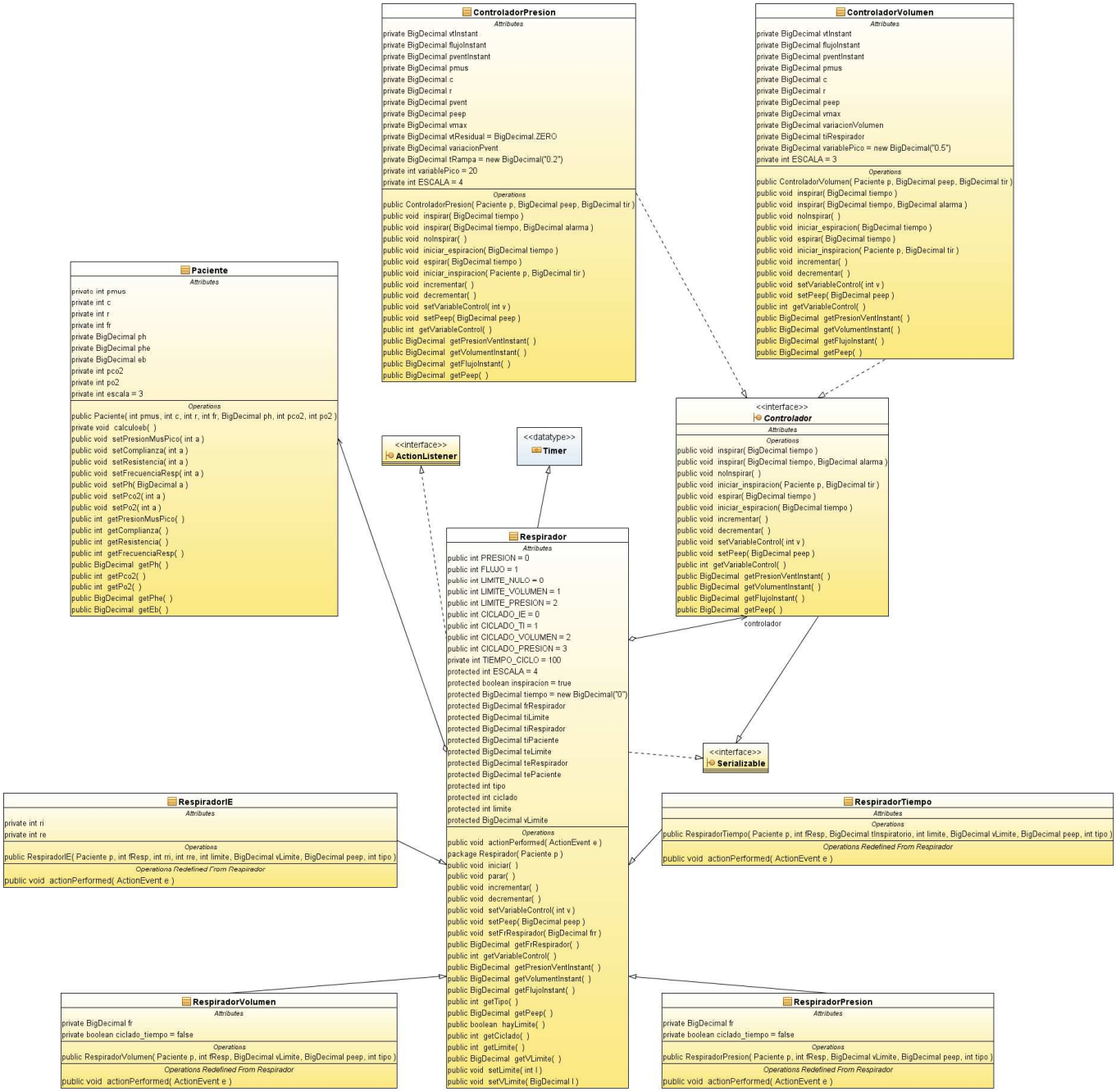


Figura 6.2.2 Diagrama de clases: Respirador

6.2.4.1. La clase Respirador

La simulación es mantenida y controlada por las clases que heredan de Respirador. Respirador implementa los métodos get y set comunes y el método de iniciar y detener la simulación, pero es en las subclases donde se mantiene el control del tiempo del ciclo respiratorio.

Las subclases sobrescriben el método actionPerformed heredado de Timer, el cual es ejecutado cada 100 milisegundos. Éste método mantiene el contador de tiempo y realiza las comprobaciones para determinar el momento del ciclo respiratorio en el que se

encuentra el paciente. Los algoritmos implementados en cada una de las subclases son muy similares, la diferencia radica en la variable o variables que determinan el paso de una fase a otra. Las funciones que se realizan en el método actionPerformed() son:

Estado de fase:

En cada iteración el contador de tiempo aumenta 0.1s (100ms) y se resetea al inicio de la inspiración y al inicio de la espiración. Seguidamente, si el ciclado del respirador es por tiempo, se compara el tiempo del contador con el tiempo inspiratorio si el flag inspiración está activo y con el tiempo espiratorio si no lo está. Según el resultado de la comprobación se llama al método del controlador correspondiente. Cada vez que se comienza un nuevo ciclo se actualiza el objeto paciente del controlador por si se hubiesen producido modificaciones en alguno de sus parámetros y el tiempo inspiratorio ajustado en el respirador.

El siguiente pseudocódigo muestra de forma sencilla la implementación:

```
Si (tiempo < T. inspiración y inspiración = true)
    ⇨ controlador.inspirar(tiempo);
Si (tiempo >= T.inspiración y inspiración = true)
    ⇨ controlador.iniciar_espiracion();
    ⇨ tiempo=0; inspiración=false;
Si (tiempo < T.espiración y inspiración = false)
    ⇨ controlador.espirar(tiempo);
Si (tiempo >= T.espiración y inspiración = false)
    ⇨ Tiempo=0; inspiración=true;
    ⇨ Recalcular tiempos inspiratorio y espiratorios
    ⇨ Controlador.iniciar_inspiración(paciente,T.inspiratorioRespirador)
```

Si el ciclado del respirador en lugar de ser por tiempo es por presión o volumen, se añade en los dos primeros “Si” la comparación del límite con el valor actual. La comparación con el tiempo inspiratorio se mantiene debido a que si no se alcanza el límite durante ese tiempo, se produce el ciclado.

Respiración controlada o asistida:

Según el requisito impuesto, el respirador debe pasar de modo controlado a modo asistido y viceversa automáticamente si detecta o no un esfuerzo inspiratorio por parte del paciente. Para ello, primero en el constructor y luego cada vez que entra en la fase de inicio de la inspiración, comprueba si la presión muscular del paciente es igual a cero. Si es igual, el paciente no realiza esfuerzo por lo que la FR (frecuencia respiratoria) que se utiliza es la establecida en el respirador; pero si es mayor que cero (no puede ser negativa), se utiliza la FR del paciente. La frecuencia respiratoria influye en el cálculo del tiempo inspiratorio y espiratorio.

Alarmas y pausas inspiratorias:

Cuando el respirador está en modo asistido, puede ocurrir que la FR del respirador sea mayor que la FR del paciente. Si esto ocurre, al ser el TI (tiempo inspiratorio) del respirador menor que el del paciente, el respirador realiza una pausa hasta que el paciente realiza el ciclado. Esto implica que deja de generar flujo, por lo tanto el volumen no aumenta y la presión cae. En el caso de que sea al contrario, el TI del respirador no se tiene en cuenta, usando únicamente el TI del paciente. Para conocer esta condición se añade en

la inspiración una comparación del tiempo con el TI del respirador y si es válida en lugar de ejecutar controlador.inspirar(tiempo) ejecuta controlador.noinspirar().

En el caso de las alarmas ocurre lo mismo cuando el ciclado es por tiempo (el modo es indiferente). Si el parámetro de alarma alcanza el valor establecido, el respirador realiza una pausa hasta que termina el tiempo inspiratorio. Para conocer esta condición se comprueba en la inspiración si existe un límite activo y si lo hay ejecuta controlador.inspirar(tiempo,valor_limite), siendo el controlador el que maneja la condición del límite.

Controlador:

El controlador es instanciado en el constructor del objeto Respirador. El Respirador recibe un argumento con el tipo de controlador que debe instanciar (presión o volumen). Aunque según el tipo de respirador, alguno solo tiene la posibilidad de utilizar un tipo de controlador.

6.2.4.2. El interfaz Controlador

Una vez explicado cómo se simula el ciclo respiratorio, el siguiente paso es ver cómo se simula los cambios producidos en un paciente durante cada fase del ciclo.

La interfaz controlador, como se ha explicado anteriormente, define los métodos necesarios que deben tener las clases que la implementen y actúen de controlador para el respirador. Dejando aparte los métodos get y set, los métodos más importantes que contiene son los utilizados en cada fase del ciclo respiratorio.

En el apartado 1.3.2, se mencionó la ecuación del movimiento del sistema respiratorio. Ésta ecuación, en sí, no es una ecuación algebraica, como puede parecer, sino más bien una ecuación diferencial lineal con coeficientes constantes (compliance y resistencia).

La diferencia entre el control por volumen y el control por presión radica en la forma de resolver la ecuación y en las variables necesarias para realizarlo.

En el controlador por volumen, se establece un volumen corriente, el cual se mantendrá constante independientemente del resto de parámetros. Es decir, si variara la resistencia de un paciente, ésta no influiría en el volumen corriente si no en la presión y en el flujo los cuales se calculan a partir del volumen y el resto de parámetros. Por lo tanto, la curva de volumen y flujo inspiratorios no varía.

En el caso del controlador por presión, el objetivo también es mantener un control del volumen corriente. Sin embargo, al ser la presión el parámetro de control, el volumen corriente depende no sólo de la configuración del ventilador, sino también de la compliance y resistencia del sistema respiratorio del paciente. Por lo tanto, si variara la resistencia de un paciente, la curva de volumen y flujo inspiratorio se vería afectada.

Los siguientes diagramas representan las ideas más fundamentales de la ventilación mecánica. Para poder realizar una simulación correcta se debe comprender todas las variables y las relaciones que existen entre ellas.

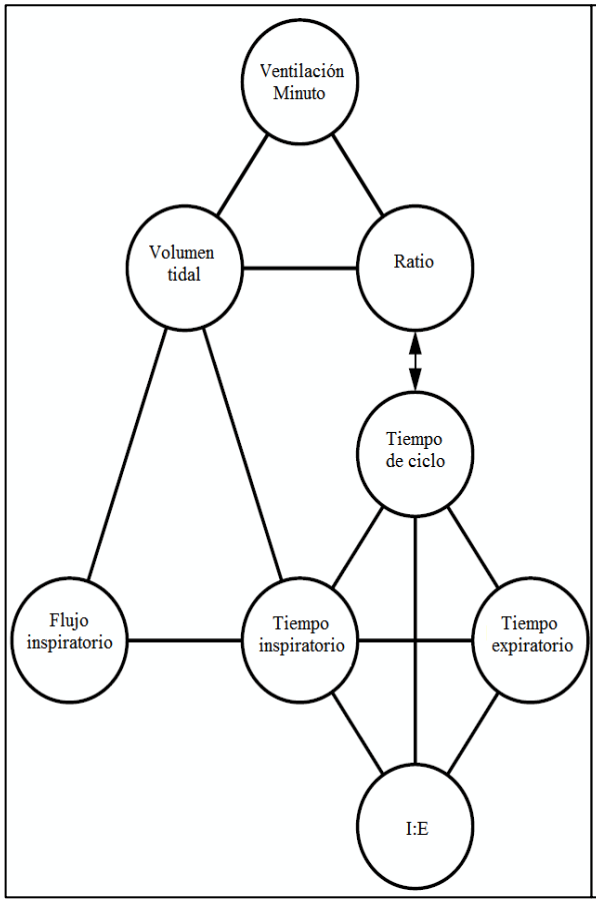


Figura 6.2.4 Diagrama de control por volumen

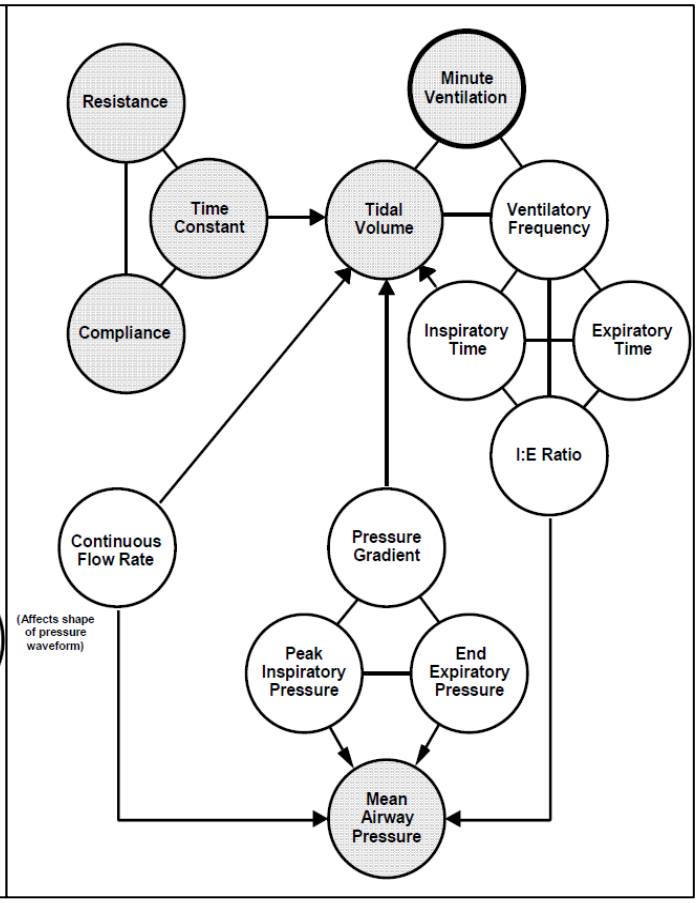


Figura 6.2.3 Diagrama de control por presión

Como se puede observar en las Figuras 6.2.3 y 6.2.4, el cálculo de los parámetros en el controlador por volumen es más sencillo que por presión. En el controlador de presión se utiliza el término de constante de tiempo (Resistencia x Complance) para realizar el cálculo del volumen, y por tanto del flujo, durante la inspiración.

Controlador de Presión

En el método iniciar_inspiración(), se actualizan los valores estáticos del paciente (compliance y resistencia) y se resetean y ajustan los parámetros variables conforme al valor con que terminaron la espiración o a los valores iniciales. Es el instante de tiempo cero el valor de los parámetros es:

- Volumen: es igual al volumen que no se ha expulsado de los pulmones al terminar la espiración (volumen residual).
- Presión ventilatoria: es igual al valor de la PEEP
- Flujo: es igual al valor final del flujo espiratorio.

El método inspirar() está sobrecargado. En un caso recibe como argumento el tiempo (contador) y en el otro el valor de la variable límite, además del tiempo. Éste método controla el aumento de presión hasta el valor de la presión pico durante el tiempo rampa establecido (0.2 s por defecto) y la mantiene constante, modifica el valor del volumen y el flujo conforme a las fórmulas de cada uno y, en el segundo caso, además controla si se ha alcanzado el valor de alarma del volumen. Si ocurre esta última condición,

se llama al método noinspirar(), el cual establece el flujo a cero y obtiene el valor de la presión conforme a la ecuación del movimiento (presión meseta). Las fórmulas utilizadas han sido calculadas a partir de la ecuación del movimiento y la constante de tiempo.

$$Vt = \Delta P \times C \times \left(1 - e^{-t/R \times C}\right)$$
$$F = \left(\frac{\Delta P}{R}\right) \times \left(e^{-t/R \times C}\right)$$
$$\Delta P = PIP - PEEP$$

En el método iniciar_espiración() se ajustan los parámetros con los valores que deben tener al inicio de la espiración.

- Presión: es igual a la PEEP.
- Volumen: igual al último valor alcanzado en la inspiración.
- Flujo: igual a cero (no hay flujo de aire ni hacia adentro ni hacia afuera).

Por último, en el método espiración() la presión se mantiene constante con el valor de la PEEP y el volumen y flujo varían en función de las fórmulas calculadas a partir de la ecuación del movimiento y la constante de tiempo.

$$Vt = \Delta P \times C \times \left(e^{-t/R \times C}\right)$$
$$F = -\left(\frac{\Delta P}{R}\right) \times \left(e^{-t/R \times C}\right)$$
$$\Delta P = Vmax/C$$

Vmax = Volumen máximo alcanzado durante la inspiración.

La relación que existe entre las fórmulas dadas y la ecuación del movimiento viene determinada por la constante de tiempo. Según se explicó en el capítulo 1, la constante de tiempo es el producto de la resistencia por la compliance y su valor indica el tiempo que necesita un pulmón para llenarse o vaciarse un 63% del volumen corriente que puede contener. Realizando los cálculos, el pulmón se llena/vacía completamente cuando el tiempo tiende a infinito, pero normalmente se considera el 100% cuando el tiempo es igual a cinco veces la constante de tiempo.

Durante la espiración, en lugar de utilizar la expresión PIP-PEEP, la cual corresponde al volumen máximo que entra los pulmones divido por la compliance, se utiliza el volumen máximo inspirado dividido por la compliance ya que en realidad éste es el valor real de volumen ingerido puesto que el tiempo puede no haberse llegado a igualar a 5 veces la constante de tiempo.

El siguiente gráfico ilustra las constantes de tiempo en la inspiración y espiración.

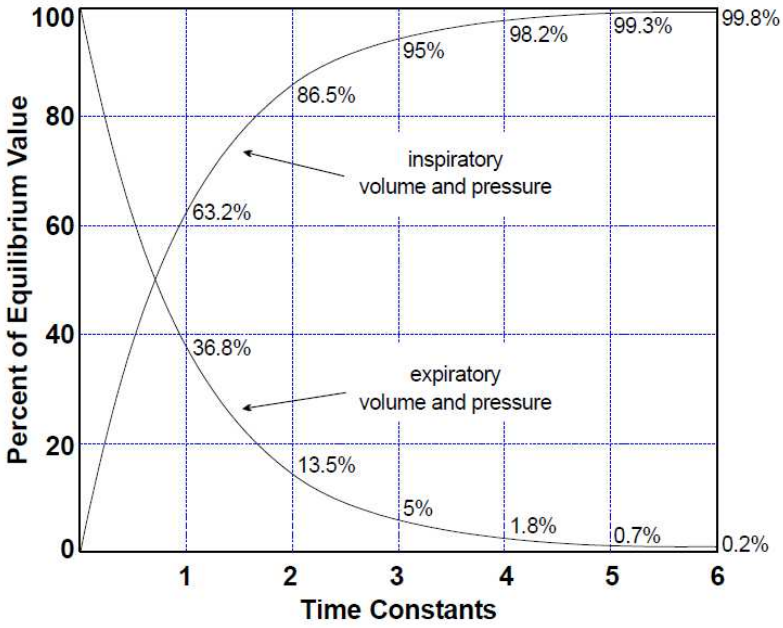


Figura 6.2.5 Gráfico de la constante de tiempo

Controlador de Volumen

El método iniciar_inspiración(), se implementa igual que en el controlador de presión. Además, se realiza el cálculo de la variación de volumen por unidad de tiempo respecto al tiempo inspiratorio del respirador, puesto que durante ese tiempo, el volumen tiene que aumentar de manera constante.

En el método inspirar(), el flujo se mantiene constante debido a que el incremento del volumen es también constante. En el caso de que la alarma de presión esté activa y se alcance el valor, el flujo cae a cero y el volumen se mantiene constante. La fórmula utilizada en este caso es la ecuación del movimiento puesto que la única incógnita es la presión ventilatoria:

$$Vt = Vt + Variación\ Vt$$
$$Pvent = \frac{Vt}{C} + F \times R + PEEP - Pmus$$

Los métodos iniciar_espiración() y espirar() se implementan igual que en el controlador de presión puesto que la espiración es pasiva y los pulmones se vacían de la misma manera.

En las siguientes Figuras, se puede observar el comportamiento de las curvas de presión, volumen y flujo correspondientes a las dos formas de control explicadas.

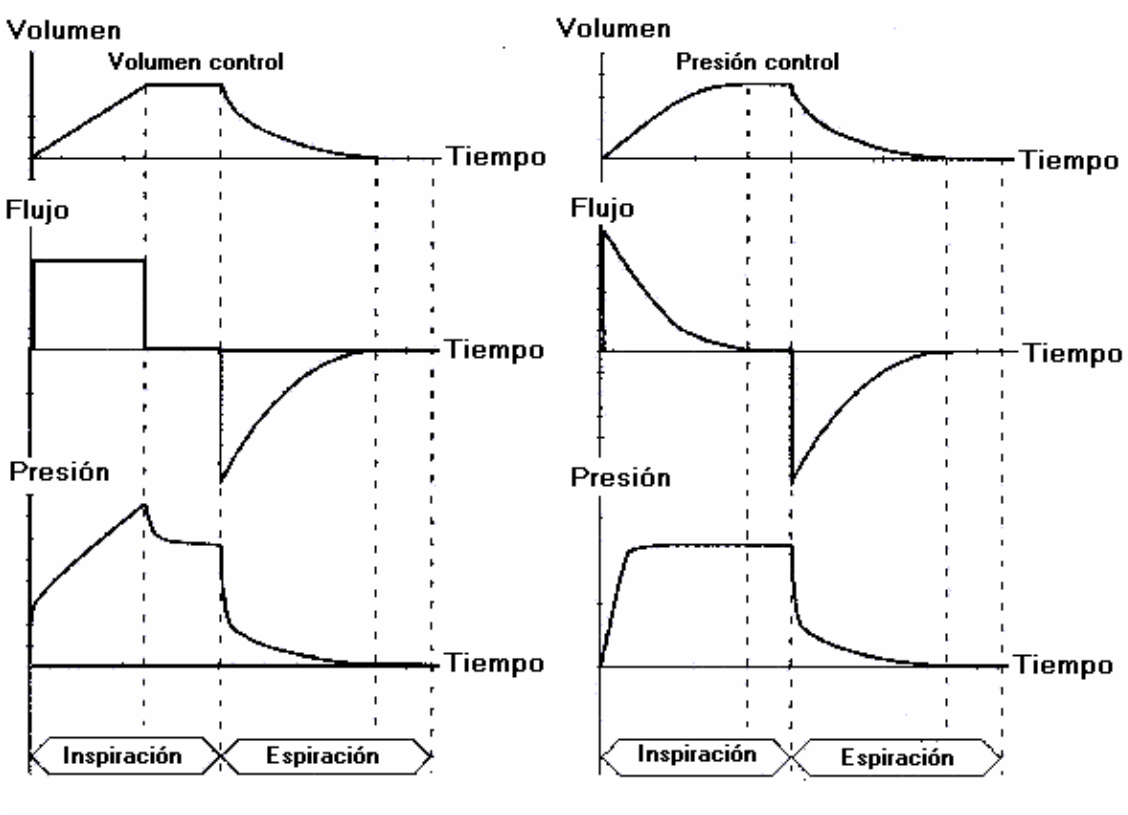


Figura 6.2.6 Gráficas de presión, volumen y flujo con pausa

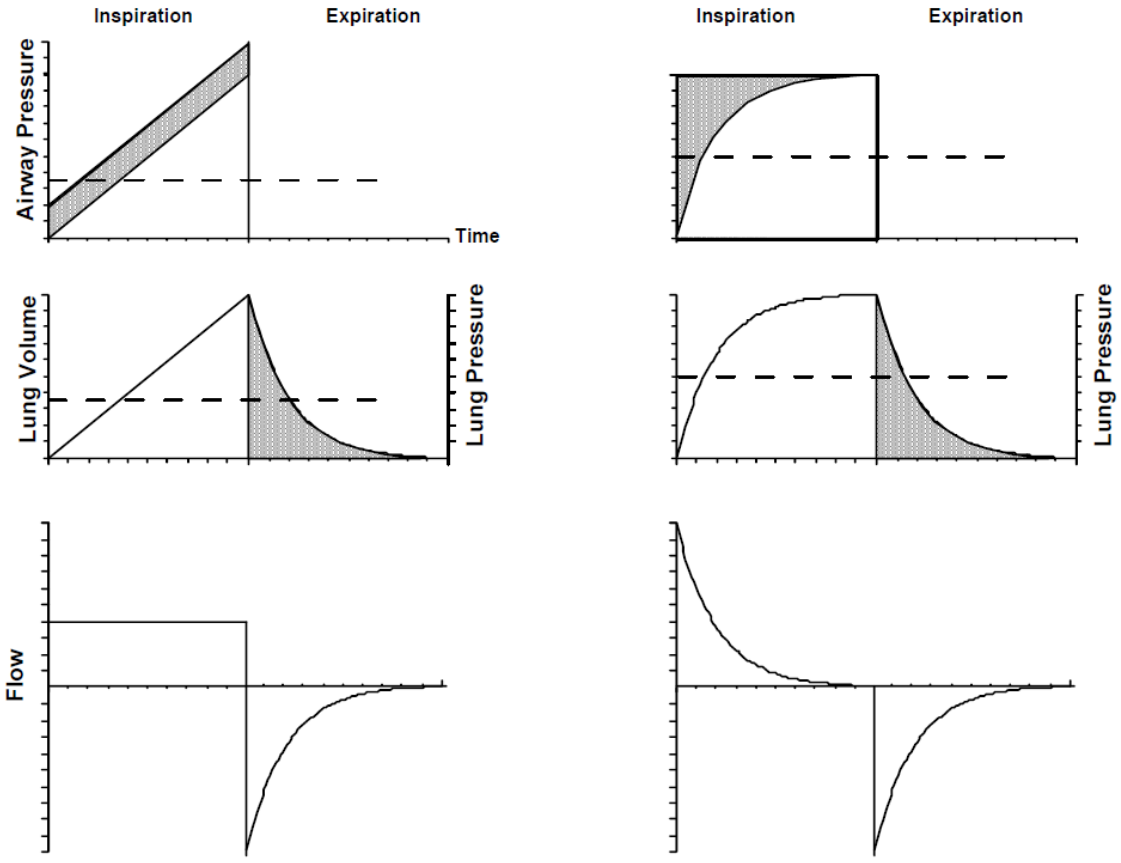


Figura 6.2.7 Gráficas de presión, volumen y flujo

6.2.4.3. La clase Paciente

La clase paciente contiene todos los atributos que representan los diferentes parámetros inherentes al paciente. Los métodos de la clase son todos los get y set para cada uno de los atributos. Se podría decir que la clase se comporta como un almacén de información ya que no realiza operaciones sobre los datos salvo en un sólo método. Este método, calculoeb(), es el que realiza los cálculos para la obtención de los resultados de la máquina de gases.

Gasometría

El objetivo de la gasometría, o máquina de gases, es la obtención del pH esperado que debería tener el paciente y el valor del equilibrio ácido base del paciente, partiendo de los valores del pH, pCO2 y pO2.

La simulación de la máquina de gases es independiente de la simulación de la mecánica respiratoria y es propia del paciente, es decir, el respirador no actúa de ninguna manera sobre éstos parámetros. Por lo tanto, el método calculoeb() es llamado únicamente cuando el usuario modifica alguno de los 3 parámetros.

Las fórmulas utilizadas para el cálculo son:

Si (pCO2 > 40)
⇨ *Variación pH = [(pCO2 - 40)/2]/100;*
Si (pCO2 > 40)
⇨ *Variación pH = (40 - pCO2)/100;*
Si (pCO2 = 40)
⇨ *Variación pH = 0;*

pH Esperado = 7,4 – Variación pH
EB = (pH paciente – pH esperado) x 200/3

Capítulo 7. CONCLUSIONES

La herramienta desarrollada en el proyecto es una aplicación real desarrollada a petición del Hospital de Navarra, concretamente por el Doctor Juan Pedro Tirapu, para simular casos prácticos de ventilación mecánica en las clases que imparte a sus compañeros. La aplicación ha sido desarrollada a medida según los requisitos impuestos por el Doctor Tirapu y otros que he considerado convenientes.

Mi principal motivación para la elección del proyecto fue la posibilidad de crear una aplicación real, útil y que además fuera utilizada con fines educativos. Además, el proyecto ha supuesto todo un reto debido a la temática en la que se fundamenta; un campo que me era totalmente desconocido y sobre el que he tenido que hacer una amplia investigación para conocer el funcionamiento del sistema que deseaba simular.

Uno de los puntos que considero más significativos ha sido la comunicación mantenida con el doctor a lo largo de todo el desarrollo del proyecto. Hemos realizado diversas reuniones tanto en el laboratorio de proyectos de la UPNA como en la UCI del Hospital de Navarra. Ha sido en estas reuniones donde he podido comprobar la dificultad que existe cuando cada parte quiere expresar sus ideas de una forma clara y directa que pueda entender la otra. A pesar de ciertas dificultades, las reuniones han sido muy positivas ya que han permitido marcar los objetivos de la aplicación y han ayudado comprender todos los conceptos de la ventilación mecánica y el funcionamiento tanto del aparato respiratorio humano como de los distintos respiradores. Considero este punto realmente importante porque me ha permitido conocer de primera mano el proceso de diálogo con el cliente (en este caso el doctor), y la importancia que tiene este proceso en la planificación y el desarrollo de aplicaciones informáticas.

Una de las grandes dificultades del proyecto ha sido la búsqueda de las fórmulas a utilizar para la simulación debido a que la mayoría de la documentación existente está orientada a personal médico y no suelen tratar temas tan físicos ni matemáticos de funcionamiento. En todos los recursos encontrados se explica en qué consiste la ventilación mecánica, qué tipos de respiradores existen, en qué modelo matemático se basan (ecuación del movimiento), qué efectos producen su utilización… pero en apenas un par se habla de cómo funciona realmente el respirador internamente, cómo se van modificando los diferentes parámetros en función del tiempo… Esto ha supuesto uno de

los principales obstáculos que más ha retrasado el desarrollo del proyecto pero que finalmente ha sido superado.

Uno de los puntos satisfactorios del proyecto ha sido la evolución, en parte, de los requisitos iniciales. Al comienzo del proyecto, los requisitos marcados fueron simular un aparato de ventilación mecánica no invasiva de la modalidad BiPAP el cual funciona mediante control de presión, en modo asistido y realizando el ciclado por tiempo. Al comenzar el diseño de la arquitectura, uno de mis objetivos fue realizarla lo más modular posible, de forma que pudiera ser ampliada en un futuro sin demasiados problemas. Por ello, en una de las reuniones se decidió ampliar los requisitos iniciales y añadir más características disponibles en los respiradores comerciales como fueron el controlador de volumen, el ciclado por presión y volumen, las alarmas… dando la posibilidad de crear una simulación personalizada con diferentes configuraciones. Otro de los requisitos impuestos con posterioridad fue la implementación de la máquina de gases.

En general, ha sido un proyecto muy interesante, tanto la etapa de análisis e investigación como la de desarrollo e implementación han resultado ser muy instructivas. Se han cumplido los objetivos marcados y la aplicación ha sido entregada al doctor J.P. Tirapu quién ha comprobado su correcto funcionamiento.

7.1. Líneas futuras

Una vez terminada la aplicación, en estos momentos no existen más necesidades por parte del doctor Juan Pedro Tirapu, aunque la aplicación podría ser mejorada o se le podrían añadir más funcionalidades.

La interfaz gráfica podría ser mejorada, tomando un aspecto más profesional. También se podría utilizar un motor gráfico 3D para la realización y representación de la simulación, incluso se podrían añadir unas escenas en 3D interactivas con los pasos a seguir para conectar a un paciente a un respirador.

Si en un futuro se desea poder realizar exámenes a alumnos a través de la aplicación se podría incluir un sistema de gestión de usuarios conectado a una base de datos donde se almacenarían las estadísticas y resultados de la simulación.

Aprovechándose de la modularidad de la aplicación se podrían implementar además más modalidades de funcionamiento de respiradores si éstas fueran necesarias.

Estos podrían ser algunos de los puntos en los que se podría trabajar en un futuro para aumentar la funcionalidad de la aplicación. Aunque también se podrían mejorar las ya existentes.

Referencias

Ventilación mecánica

- **Páginas webs más significativas**
  - http://bvs.sld.cu/revistas/mie/vol1_1_02/miesu102.htm
  - http://www.enfermeriarespira.es/
  - http://www.eccpn.aibarra.org/temario/temario.htm
  - http://www.aibarra.org/ucip/
  - http://ingmarmedical.com/index.html
  - http://www.aibarra.org/Apuntes/criticos/#Cardiovascular_y_Respiratorio
  - http://www.medynet.com/usuarios/jraguilar/Manual%20de%20urgencias%20y%20Emergencias/ventmeca.pdf
  - http://redalyc.uaemex.mx/redalyc/pdf/304/30401004.pdf
  - http://www.rcjournal.com/contents/03.07/03.07.0301.pdf

- **Libros**
  - Fundamentals of Mechanical Ventilation: A Short Course on the Theory and Application of Mechanical Ventilators.
    Autor: Robert L. Chatburn.
  - Ventilación mecánica: conocimientos básicos.
    Autores: Ángeli Armes, Mª Rosario Mosegue y Miriam Galloway.
    Web:http://www.minsa.gob.ni/enfermeria/doc_inter/vent_mecanic_princ_basic.pdf
  - Principles & Practice of Mechanical Ventilation. Second edition.
    Autor: Martin J. Tobin.

Documentación Java

- **General**
  - **http://www.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/Java/Java2.pdf**

- **JFreeChart**
  - **http://www.jfree.org/index.html**

- **JavaDoc**
  - **http://www.javahispano.org/contenidos/es/documentar_con_javadoc/**
  - **http://informatica.uv.es/iiguia/LP/laboratorio/P1/p1.pdf**

- **Instalador**
  - **http://izpack.org/**
  - **http://launch4j.sourceforge.net/**

Anexo

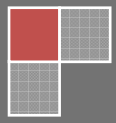
- **Manual de usuario**
- **Documentación JavaDoc del código fuente.**

2010

VMSof
Manual de usuario

Simulador de ventilación mecánica

Daniel Ventura Morales



ÍNDICE

INTRODUCCIÓN

VMSoft es una aplicación informática para simular la utilización práctica de un aparato de ventilación mecánica sobre una persona con insuficiencia respiratoria.

La aplicación simula la mecánica respiratoria de un paciente y la interacción que produce sobre ésta el uso de una máquina de respiración artificial. En la simulación se encuentran abarcados diferentes modos de ventilación existentes, pudiendo recrear diferentes casos clínicos configurables.

La simulación se realiza en tiempo real mediante la configuración previa (manual o por defecto) de las diferentes variables del paciente y del respirador.

El objetivo que se persigue con la aplicación del presente manual es:

- Dar a conocer a los usuarios finales las características y las formas de funcionamiento del software VMSof.

REQUISITOS DEL SISTEMA

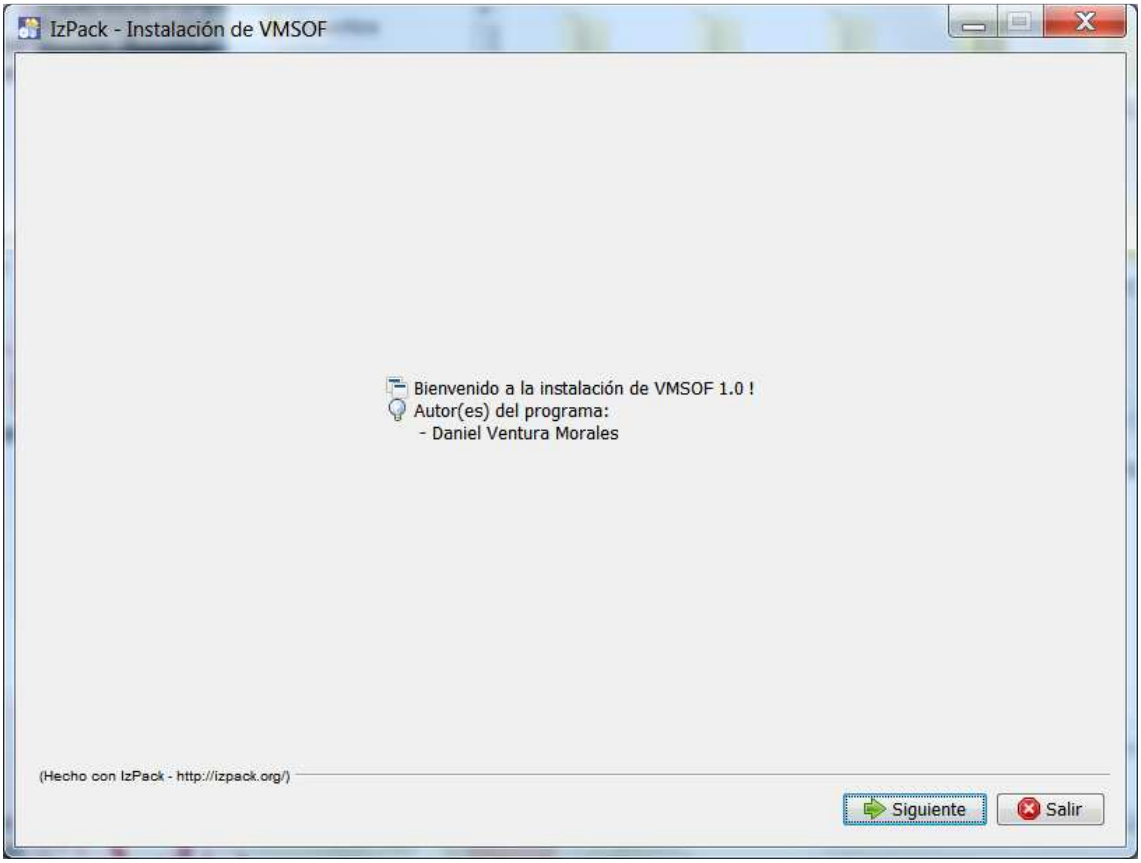
- La aplicación podrá ser instalada en cualquier ordenador de sobremesa o portátil con un hardware básico. Se recomienda:
  - Procesador Pentium IV o superior.
  - 1024 MB de memoria RAM.
  - 10 MB de espacio en disco disponible.
  - Sistema Operativo Windows 2000 o superior.

- Es necesario tener instalada la máquina virtual de Java (JRE 6.0 o superior) para poder instalar la aplicación. Si no tiene instalado el JRE puede descargarlo de la página oficial de Java SUN Microsystems: http://www.java.com/es/download/index.jsp

INSTALACIÓN

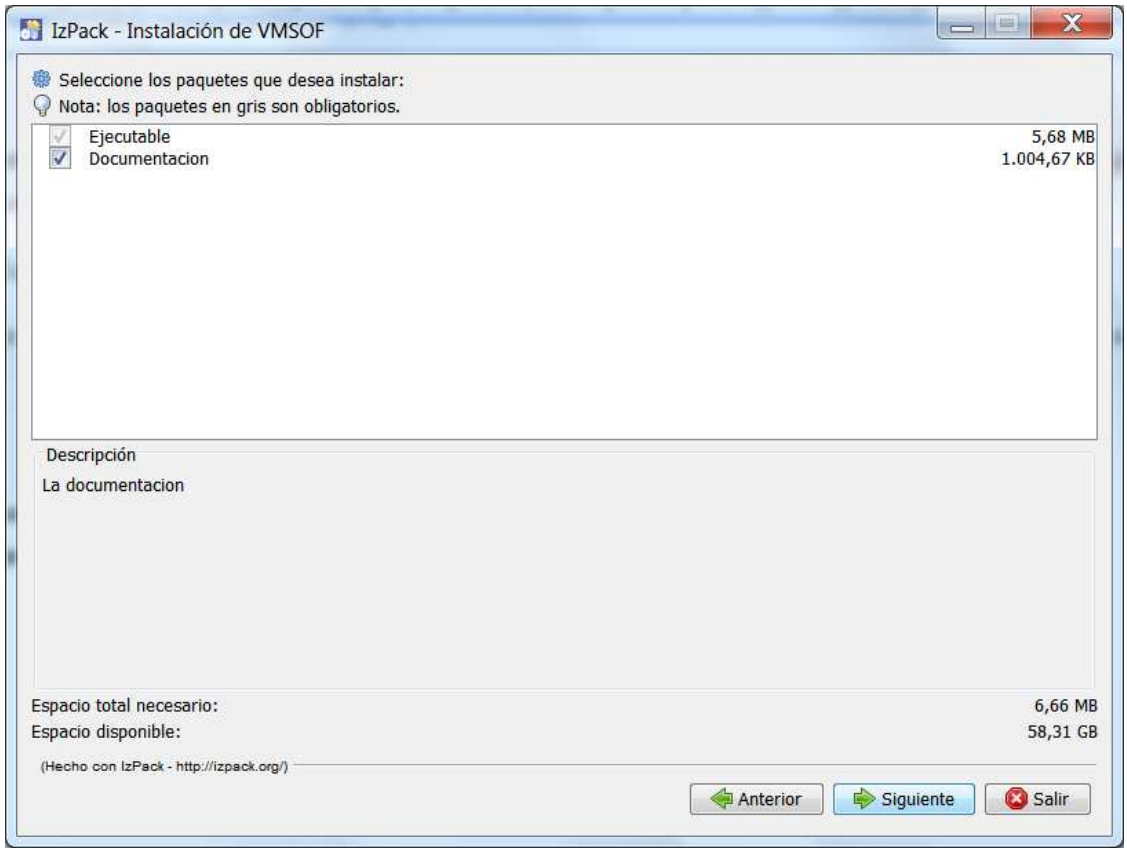
La aplicación cuenta con dos instaladores, vmsof_install.exe y vmsof_install.jar. El instalador en formato .exe es exclusivo para sistemas operativos Windows mientras que el .jar puede ser utilizado además en sistemas operativos Linux.

Al ejecutar cualquiera de los dos se despliega la siguiente ventana.

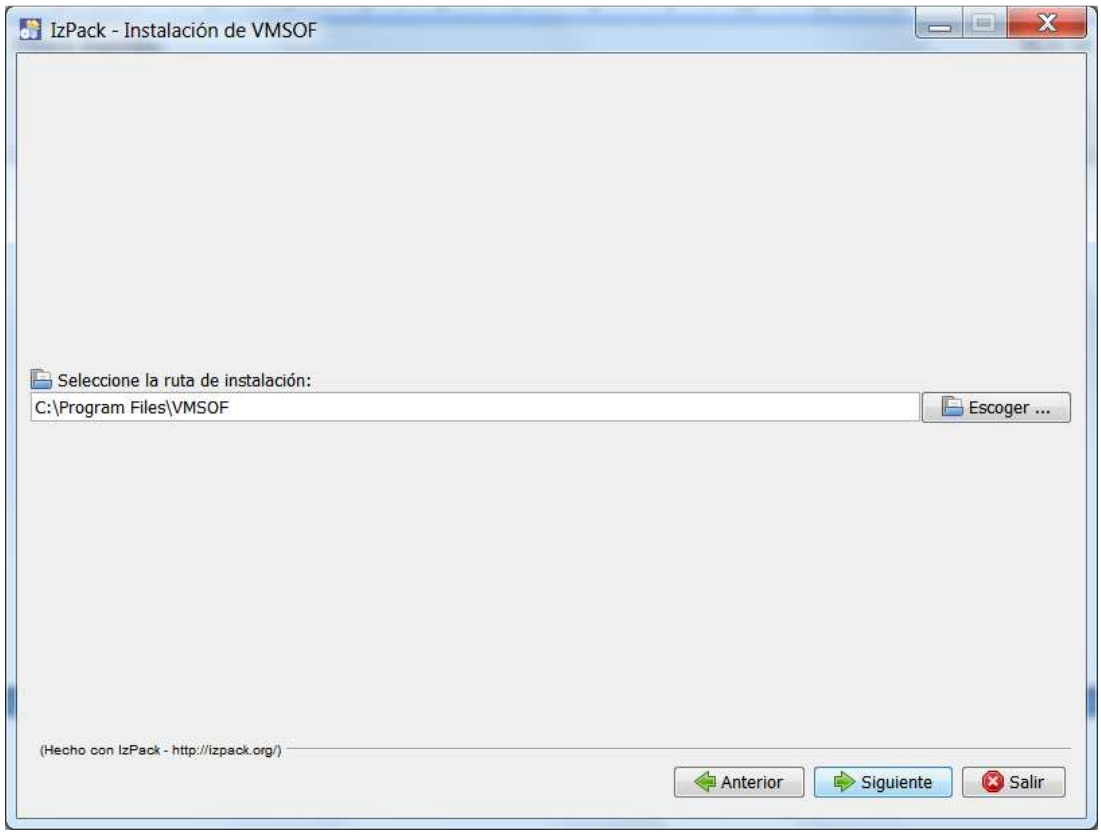


** Si utiliza el sistema operativo Windows Vista o Windows 7 deberá proporcionar permisos de administrador cuando se le solicite.*

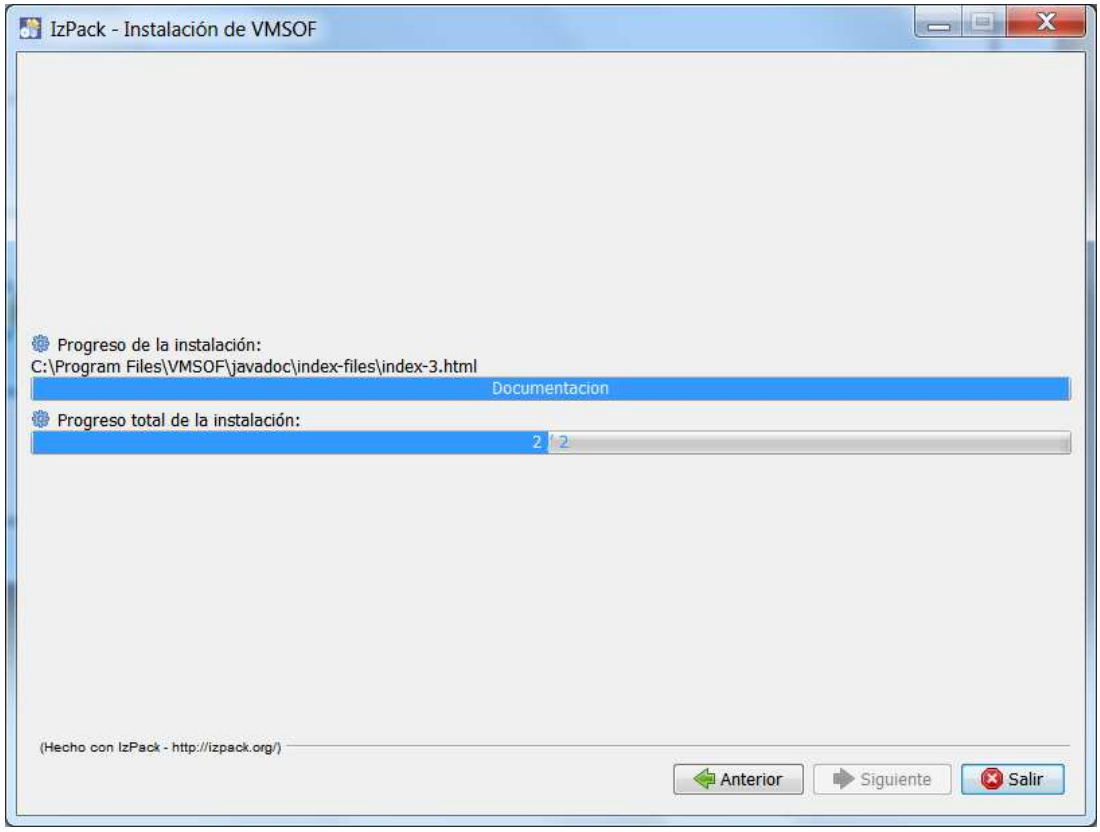
En la siguiente ventana podrá elegir si desea instalar la documentación del código fuente asociada a la aplicación en formato HTML. La instalación básica, que incluye los ejecutables de la aplicación y los archivos necesarios, es obligatoria.



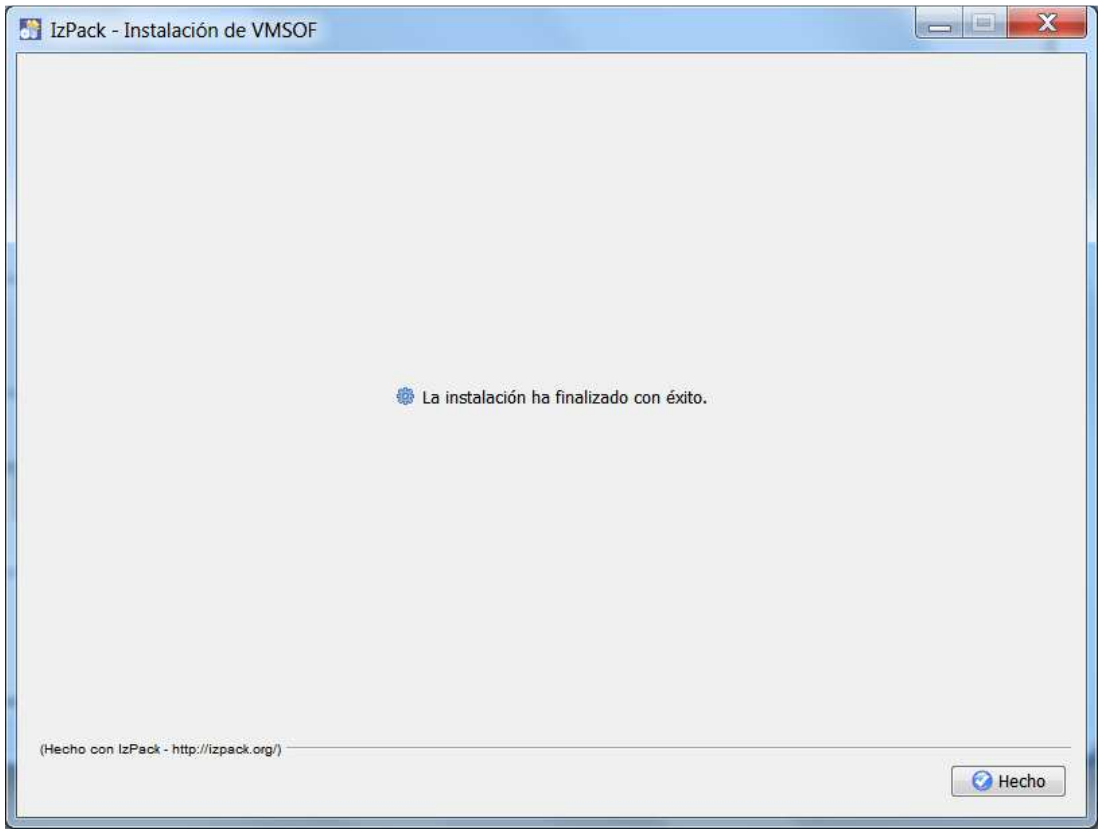
En esta ventana podrá definir la ruta de instalación de la aplicación.



Espere unos instantes hasta que termine la instalación y pulse siguiente.



Por último, pulse hecho para completar la instalación.



FUNCIONAMIENTO

VMSof cuenta con varias funcionalidadesque se irán explicando a lo largo de este apartado del manual.

Menú de inicio

Al iniciar la aplicación el primer formulario que aparece es el menú de inicio. Desde este menú puede acceder a las funciones de crear una nueva simulación y a la de cargar una simulación almacenada previamente. Además tiene un botón en la parte inferior que le permite cerrar la aplicación.

1 – Abrir ventana para configurar una nueva simulación.

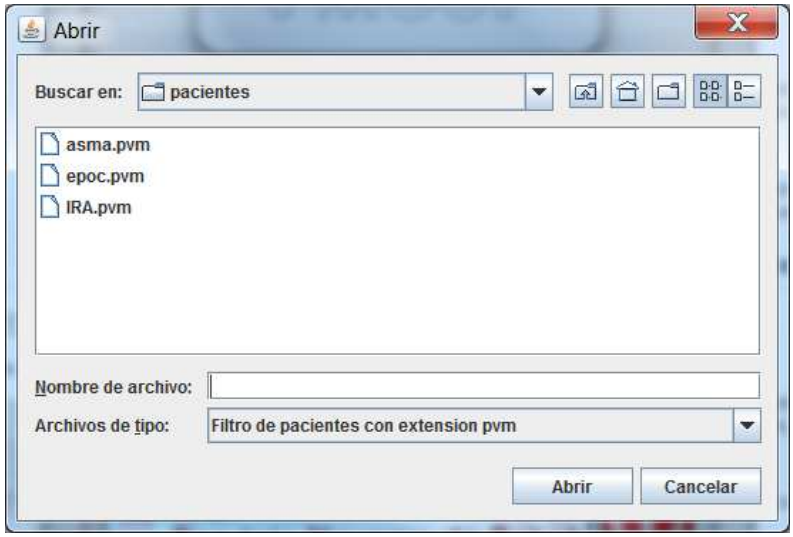
2 – Cargar una simulación almacenada.

3 – Salir de la aplicación

Cargar una simulación

VMSof admite ficheros .pvm exclusivos de la aplicación. Si el archivo elegido es incorrecto se le mostrará un mensaje de error y podrá seleccionar otro nuevamente.

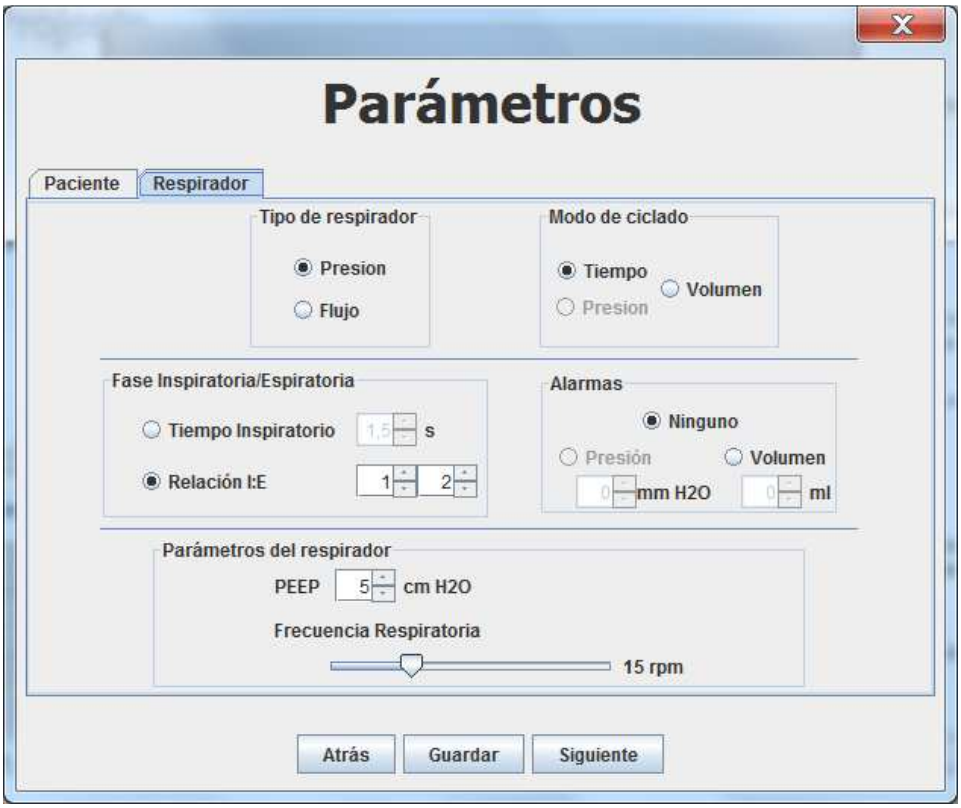
Al cargar un fichero correcto se abrirá la ventana de simulación para poder comenzar su ejecución.



Menú de configuración

En el menú de configuración puede configurar todos los parámetros disponibles del paciente y del respirador.

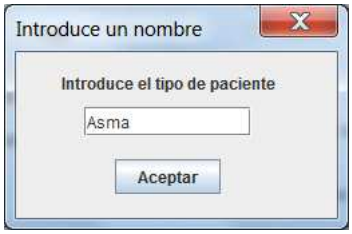
En la pestaña de paciente, los parámetros se encuentran divididos en dos grupos: los parámetros que influyen en la simulación de la ventilación mecánica y los parámetros que pertenecen a la simulación de la máquina de gases.



En la pestaña del respirador, la ventana se divide en tres líneas, las dos primeras permiten elegir la modalidad del respirador mientras que la última son los parámetros regulables durante la simulación. Como se aprecia en la imagen y durante el uso de la aplicación, no todas las opciones de configuración están disponibles para todas las modalidades de funcionamiento. Por ejemplo, si está activo el tipo presión, el modo de ciclado por presión esta deshabilitado debido a que son incompatibles.

Por último, los botones de navegación de la parte inferior, permiten volver al menú de inicio, guardar la configuración de la simulación y crear los objetos de simulación. Al terminar las dos últimas operaciones, se abre el menú de simulación.

Al elegir la opción de guardar se le solicita un nombre para el nuevo fichero. Si existe un fichero con el mismo nombre se le preguntará si desea sobrescribirlo. Los ficheros se almacenan en la carpeta pacientes en la ruta donde se ha realizado la instalación. Si la carpeta no existe la aplicación la creará y si no tiene permiso se le solicitará que la cree manualmente.

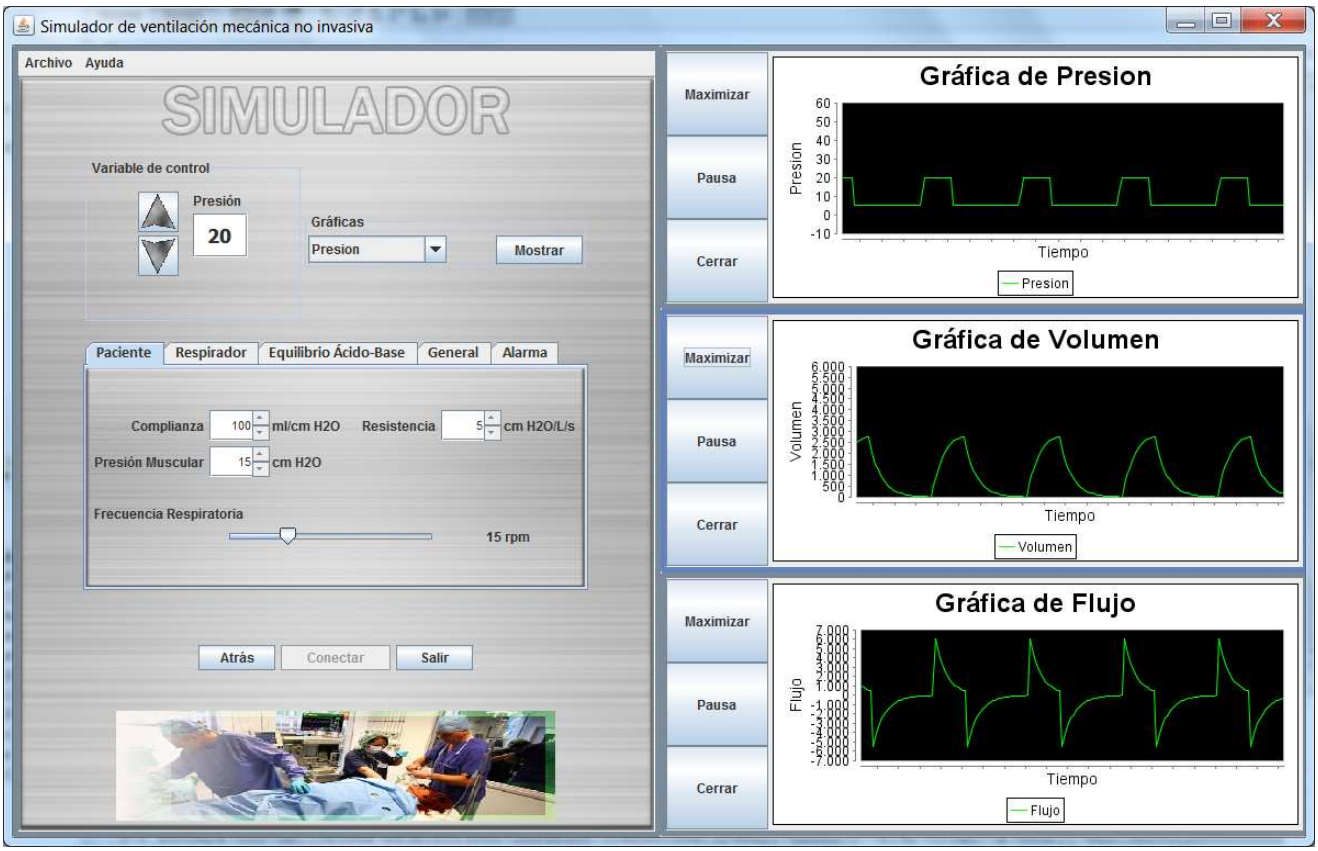


Menú de simulación

En el menú de simulación se muestran los datos obtenidos durante el proceso de simulación y permite modificar los parámetros del respirador y del paciente durante la simulación.

En la parte izquierda se encuentra el panel de configuración. En el centro del panel, puede modificar el valor de los parámetros de la simulación, los cuales se encuentran ordenados en diferentes pestañas según su naturaleza. En la parte superior se encuentra la lista desplegable con las diferentes gráficas disponibles junto con el botón para mostrarlas y la variable de control del respirador (presión o volumen según el tipo de respirador elegido). Y en la parte inferior, se encuentran los botones para volver al menú de inicio, salir de la aplicación y comenzar la simulación.

La parte derecha de la ventana contiene el panel de las gráficas. Este panel puede contener hasta 3 gráficas simultáneamente, cada una de ellas con sus respectivos botones para activar o desactivar las diferentes funcionalidades.



Funciones de las gráficas:

- Cerrar: Cierra la gráfica elegida.
- Pausa/Reanudar: Detiene o inicia la gráfica elegida.
- Maximizar/Restaurar: Maximiza o restaura el tamaño original la gráfica elegida

- Zoom: Seleccionar con el botón izquierdo del ratón el área de la gráfica hacia la derecha donde se desea hacer zoom. Para restaurar pulsar con el botón izquierdo sobre la gráfica y arrastrar hacia la izquierda.
- Mostrar: Abre la gráfica elegida.

SIMULACIÓN

VMSof permite simular los siguientes modos de ventilación mecánica:

Modos

- Asistido: El respirador entra en modo asistido automáticamente cuando el paciente realiza un esfuerzo inspiratorio, es decir, cuando la presión muscular es mayor que cero. En este caso, el tiempo inspiratorio del ciclo es el marcado por el paciente.
- Controlado: El respirador entra en modo controlado automáticamente cuando el paciente no realiza ningún esfuerzo inspiratorio, es decir, cuando la presión muscular es igual a cero. En este caso, el tiempo inspiratorio del ciclo es el marcado por el respirador.

Tipos

- Presión: Simula un respirador controlado por presión.
- Volumen: Simula un respirador controlado por volumen.

Ciclado

- Tiempo: Simula un respirador que cicla por tiempo inspiratorio. El tiempo inspiratorio se puede ajustar de dos formas:
  - Especificando un valor para el tiempo inspiratorio
  - Especificando el valor de la relación I:E.
- Volumen: Simula un respirador que cicla por volumen. Éste sólo es compatible con el respirador de tipo presión. El valor se ajusta en el apartado de Alarmas en la pestaña del Respirador.
- Presión: Simula un respirador que cicla por presión. Éste sólo es compatible con el respirador de tipo volumen. El valor se ajusta en el apartado de Alarmas en la pestaña del Respirador.

Alarmas

- Presión: Simula una condición de alarma cuando el respirador alcanza el nivel de presión indicado y produce una pausa hasta que termina el tiempo inspiratorio. Esta alarma es compatible con el modo de ciclado por tiempo en respiradores de tipo volumen.
- Volumen: Simula una condición de alarma cuando el respirador alcanza el nivel de volumen indicado y produce una pausa hasta que termina el tiempo inspiratorio. Esta alarma es compatible con el modo de ciclado por tiempo en respiradores de tipo presión.