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

Título: Sistema de configuraciones dinámicas y ambientes virtuales para el Rehabilitador de Marcha.

Autor: Jorge Raydel Sánchez Garrote

Tutores: Ing. Andrie Pérez Villamil

Ing. Alejandro Ravelo Julian

Universidad de las Ciencias Informáticas

Facultad 6

La Habana, Junio 2017

"La medida de la inteligencia es la capacidad de cambiar."

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas para que haga el uso que estime pertinente con él.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2017.

Jorge Raydel Sánchez Garrote

_____

Firma del Autor

Ing. Andrie Pérez Villamil

_____

Firma del Tutor

Ing. Alejandro Ravelo Julian

_____

Firma del Tutor

Datos de Contacto

Autor:

Jorge Raydel Sánchez Garrote

Universidad de las Ciencias Informáticas

La Habana, Cuba

Correo: jrsanchez@estudiantes.uci.cu

Tutor:

Ing. Andrie Pérez Villamil

Universidad de las Ciencias Informáticas

La Habana, Cuba

Correo: avillamil@uci.cu

Tutor:

Ing. Alejandro Ravelo Julian

Universidad de las Ciencias Informáticas

La Habana, Cuba

Correo: aravelo@uci.cu

Por siempre creer en mí en todo momento sin importar las adversidades, por depositarme toda su confianza, por sus fuerzas, sus deseos, su ímpetu a la hora de educarme, por hacerme desde muy pequeño, cada día, una mejor persona, por su cariño, por su comprensión en momentos incomprensibles, por ser la mejor madre de este mundo en creces, por ser mi todo, mis fuerzas, mis deseos, mis ganas de seguir adelante, por enseñarme a luchar con ambición por mis metas y por apoyarme en cada una de ellas por más alocadas que fuesen, por cada noche sin dormir que le brindé, que no fueron pocas en toda la carrera, por su sufrimiento y por entregarme toda su vida, dedico este trabajo de diploma a mi madre Rosa pues considero que es lo menos que puedo hacer para agradecerle todo lo que ha logrado en mí con su esmero, más que mío esto es un logro de ella, es ella quien se hace ingeniera hoy, pues si no fuese por sus miles de intentos de levantarme los ánimos y de colocarme una sonrisa en el rostro ante un bache de la vida no hubiese sido posible, fue ella quien me dio la mano desde mis inicios escolares hasta el día de hoy para junto a mí disfrutar esta meta. Luego de alcanzar este objetivo eres tú la persona que me viene a la mente con más fuerzas, tú eres la razón de cada uno de mis logros, gracias por ser tú siempre.

Hoy es un día muy especial para mí porque he alcanzado uno de los sueños de mi vida. Para el logro del mismo, muchas personas me han apoyado y brindado su ayuda, este es el momento de agradecerles a todos.

A mi madre Rosa que durante estos años de carrera estuvo a mi lado apoyándome en todo, que dio todo de sí y luchó junto a mí, día a día de esta larga travesía, que se sacrificó sin pensarlo para darme comodidades que muchas veces no merecía y se preocupó de mi bienestar mientras estudiaba, que dejó de dormir cada día de mis pruebas y sufrió en cada bache que existió en estos años. Gracias por existir, eres la persona que me dio fuerzas para alcanzar esta meta.

A mi hermana Raysa que también me apoyó mucho anímicamente en cada paso de mi carrera y me inspiró con su ejemplo y sus fuerzas a ser cada día mejor y luchar por alcanzar este logro.

A mi esposa Jany que ha sido realmente mi todo en cada momento difícil por el que cursé para lograr esta meta, que ha sabido sobrellevarme en momentos de estrés y reír conmigo cada alegría de este largo proceso, gracias por ser esa persona que eres que resume simplemente la felicidad en mi vida y los deseos de mirar cada día más lejos y tener fuerzas para emprender el viaje a cada sueño como este.

A mi tutor Andrie que estuvo al tanto día a día de mis dudas, inquietudes, que me guió por los mejores caminos a transitar en aras de lograr los más altos resultados, gracias por ser tan exigente y no perderme ni pie ni pisada. Gracias por tu dedicación, tu paciencia y tu entrega.

A mi querido amigo Manuel Ernesto que fue una pieza clave en toda mi carrera, fue el padre que no tuve para regañarme, aconsejarme, apoyarme a pesar de mis tropiezos, realmente aunque él no lo sabe le debo mucho, y lo estimo como a muy pocos en mi vida, gracias de corazón por estar ahí en momentos importantes de toda mi vida de estudiante.

A mis cuñados Daniel y Johan, a mi segunda mamá Gina por apoyarme en esta recta final, a mis grandes amigos, Rogelio, Henry y Danny por su preocupación y apoyo y a todas aquellas personas que de una forma u otra han contribuido a que este trabajo de diploma fuese posible.

Resumen

En el centro de Entornos Interactivos 3D (VERTEX) de la Universidad de las Ciencias Informáticas (UCI), se ha desarrollado una aplicación para las terapias de pacientes con problemas físico-motores, que recibe por nombre Rehabilitador de Marcha (RM) y permite al usuario médico especialista visualizar y administrar la marcha de los pacientes en la computadora. Sin embargo, este sistema presenta ciertas limitantes, por ejemplo: no cuenta con entornos virtuales para la marcha del paciente, no se pueden insertar obstáculos en la marcha, entre otras.

El objetivo de este trabajo de diploma es desarrollar entornos virtuales para el Rehabilitador de Marcha que permitan reajustar los parámetros de la marcha en relación con la posición de objetos colocados en las escenas. En aras de alcanzar dicha meta, se realizó el estudio del arte, se definieron los requisitos que debía cumplir el nuevo sistema y se documentó todo lo referente a su análisis, diseño, implementación y prueba, utilizando la metodología Extreme Programming. De esta investigación se obtuvo como resultado un componente que se integró al Rehabilitador de Marcha y permite a los pacientes realizar la terapia usando entornos virtuales y obstáculos, además de reajustar la marcha a partir de la posición de estos.

Palabras Claves: Rehabilitador de Marcha, entornos virtuales, obstáculos, pacientes, médico.

Summary

An application for the therapies of patients with physico-motor problems has been developed at the Center for 3D Interactive Environments (VERTEX) of the Informatic Sciences University (UCI). It is called Rehabilitator of March (RM) and allows the specialist medical user to visualize and to manage the march of patients through the computer. However, this system has some limitations, for example: it does not have virtual environments for the therapy of patients; it does not allow to insert obstacles in the march, and others.

The objective of this diploma work is to develop virtual environments for the Rehabilitator of March that allows readjusting the parameters of the march in relation to the position of objects placed in the scenes. In order to achieve this goal, the art study was carried out, the requirements for the new system were defined and the analysis, design, implementation and testing were documented using the Extreme Programming methodology. This investigation resulted in a component that was integrated into the Rehabilitator of March and it allows patients to perform the therapy using virtual environments and obstacles, as well as readjust the march from the position of these.

Key Words: Rehabilitator of March, virtual environments, obstacles, patients, medical.

Índice

Índice de Figuras

Índice de Tablas

Introducción

A lo largo de los últimos años, el avance de las tecnologías de la información y las comunicaciones (TICs) ha supuesto un profundo cambio en diferentes ámbitos de la vida, tanto que hoy resulta difícil imaginar cómo sería ésta sin ordenadores o teléfonos móviles, por ejemplo. Dentro del campo de la medicina, el cambio inducido por la incorporación de las TICs no ha sido menor, ya que éstas se han aplicado con fines tan diversos como la historia clínica electrónica, la tele-rehabilitación o, como es el caso que da lugar a este trabajo, la rehabilitación mediante realidad virtual (RV).

La RV resulta de gran utilidad a múltiples escalas en este sector. Existen tres ideas fundamentales referentes a ella: **interacción, involucramiento** e **inmersión** La interacción permite que el usuario interactúe con el ambiente, controlando el ritmo de trabajo. El involucramiento se obtiene a través del estímulo de los sentidos humanos (tacto, visión y audición) y la sensación de inmersión es alcanzada a través del empleo de una tecnología específica y de dispositivos como cascos de visualización o de rastreo, guantes electrónicos y palancas de mando que permiten al usuario navegar a través del ambiente virtual e interactuar con objetos virtuales. Aunque pareciera que la RV es un campo del conocimiento reciente, su historia se remonta a los años 50, de forma paralela a la invención de las primeras computadoras; no siendo así para el uso de la RV como mecanismo terapéutico, el cual es más reciente, y solo 15 años atrás la neuropsicología contempló la posibilidad de aplicar entornos virtuales en personas con diversas disfunciones del sistema nervioso central (parálisis cerebral, daño cerebral por traumatismo y trastorno de déficit atencional, entre otras). Actualmente, uno de sus campos de aplicación consiste en proponer alternativas de solución a problemas de diversidad funcional. Esta tecnología emerge como una terapia adicional que promete ayudar a las personas a superar traumas físicos o cognitivos sin enfrentar la frustración que les produce el contacto directo con la realidad. Desde esta perspectiva, la RV promete simular situaciones y espacios donde las personas pueden evaluar sus competencias y repetirlas un sin número de veces hasta lograr la perfección.

La aplicación de la RV y las nuevas tecnologías a la rehabilitación de pacientes ha originado un área novedosa: la rehabilitación virtual (RVi), poniendo a disposición del mercado internacional aplicaciones utilizadas para el apoyo a los procesos de rehabilitación las cuales se encuentran, en su mayoría, lejos del alcance de los centros rehabilitadores de Cuba por su alto costo. Debido a esto, el Centro de Rehabilitación Nacional Julio Díaz de La Habana y el Centro de Entornos Interactivos 3D (VERTEX) de la Universidad de las Ciencias Informáticas (UCI), han integrado sus esfuerzos en la realización de algunas de estas aplicaciones. Danzo Terapia (DT) y Rehabilitador de Marcha, entrenador de funciones

motoras (RM), son dos ejemplos surgidos bajo esa cooperación, además de otros proyectos muy importantes como lo es la Arquitectura de los Laboratorios Virtuales (LV).

Luego de haber liberado la primera versión del RM, el colectivo de especialistas se percató de que el sistema carece de determinadas funcionalidades imprescindibles para la realización de la terapia de los pacientes a un nivel más alto y para el logro de resultados mayores en la rehabilitación de los mismos como son: la inclusión de entornos tridimensionales donde las imágenes de los pasos sean proyectadas, la inclusión de sonido ambiente de dichos entornos, la inserción de objetos en la escena y la posibilidad de reajustar la marcha según la posición de los mismos. Estas mejoras proporcionarían una terapia más atractiva para los pacientes, al mismo tiempo que una mejor utilización del software por parte de los médicos especialistas, lográndose mayor motivación hacia el proceso de rehabilitación.

Teniendo en cuenta la situación antes descrita se plantea como **problema de la investigación:** ¿Cómo contribuir al desempeño y motivación de los pacientes durante las terapias de rehabilitación utilizando el sistema Rehabilitador de Marcha?

Se define como **objeto de estudio** las aplicaciones de rehabilitación virtual y como **campo de acción**: Los entornos tridimensionales para las aplicaciones de rehabilitación.

Para dar solución al problema científico de la presente investigación se plantea el siguiente **objetivo general**: Desarrollar e incluir entornos virtuales para el Rehabilitador de Marcha que permitan reajustar los parámetros de la marcha en relación con las características de objetos colocados en las escenas.

Para dar cumplimiento al objetivo general se define una serie de **tareas de investigación**:

1. Investigación y análisis de los sistemas empleados en los procesos de rehabilitación de pacientes que utilizan entornos virtuales.
2. Caracterización del sistema Rehabilitador de Marcha.
3. Estudio de las tecnologías más adecuadas para el desarrollo de entornos virtuales y el reajuste de los parámetros de la marcha en relación con la posición de objetos colocados en las escenas para el Rehabilitador de Marcha.
4. Diseño e integración de los entornos 3D al RM.
5. Diseño e implementación del componente que permita la inclusión de entornos virtuales y el reajuste de parámetros de la marcha en relación con la posición de los objetos en la escena para el RM.

6. Integración del componente al RM.
7. Selección de técnicas y métodos para la validación de la solución implementada.
8. Aplicación de las técnicas y métodos para validar la solución implementada.

También se planteó el problema mediante preguntas de investigación:

- ¿De qué forma se realiza la marcha en el sistema Rehabilitador de Marcha?
- ¿Qué facilidades brindan los sistemas que utilizan realidad virtual para la rehabilitación en pacientes con discapacidades físico-motoras?
- ¿Cuáles son las herramientas y tecnologías más adecuadas para el desarrollo de entornos virtuales y el reajuste de los parámetros de la marcha en relación con la posición de objetos colocados en las escenas para el Rehabilitador de Marcha?
- ¿Cuáles son las funcionalidades que debe contener el componente a desarrollar para permitir la inserción de entornos virtuales al RM y reajustar los parámetros de la marcha en relación con la posición de objetos colocados en las escenas?
- ¿Qué pruebas son más factibles aplicar para validar el nuevo componente, una vez desarrollado?

Dándole respuesta a estas preguntas se pretende lograr los siguientes resultados:

- Entornos 3D incluidos al RM.
- Componente de reajuste de la marcha para el RM.
- Documento de tesis.

La novedad de este trabajo de investigación radica en la inserción de nuevas funcionalidades al Rehabilitador de Marcha, de forma tal que mejore el desempeño de los usuarios en el mismo e intensifique su rapidez y eficacia en el ámbito de la salud. Por lo que la propuesta de nuevas funcionalidades contribuiría a un mejor desarrollo de la rehabilitación en función del proceso perceptivo-motor, constituyendo esto entre otros el aporte de la investigación.

Para concretar los componentes teóricos y prácticos antes declarados sobre las posibles aplicaciones de esta investigación y el cumplimiento de las tareas planteadas se emplearon los siguientes métodos:

Métodos teóricos:

Histórico-lógico: Este método se empleó porque permite analizar de forma lógica la trayectoria histórica de la Rehabilitación virtual, sus inicios, su evolución y desarrollo, así como los sistemas que realizan procesos similares al RM, necesarios tener en cuenta para el desarrollo del nuevo componente.

Analítico-sintético: Se seleccionó este método ya que permite dividir el objeto: las aplicaciones de rehabilitación virtual, en sus diferentes componentes: entornos virtuales, formas de realizar la marcha, objetos en las escenas, etc. De esta forma resulta más sencillo estudiar el objeto en cuestión y sintetizar los elementos necesarios para la presente investigación.

Métodos empíricos:

Observación: Se utilizó este método porque se puede usar en distintos momentos de la investigación, permitiendo observar el funcionamiento del Rehabilitador virtual en marcha con vista a la mejoría del mismo.

Modelación: Permite representar la realidad existente, modelando la necesidad actual en un diagrama de dominio que podrá sustituir al objeto de estudio definido para esta investigación. Mediante la modelación es posible obtener una mejor comprensión del objeto de estudio. Además, se emplea en el diseño del diagrama de clases.

Este documento consta de Resumen, Introducción, tres capítulos de contenido, Conclusiones, Recomendaciones, Referencias Bibliográficas, Bibliografía y Glosario de Términos. A continuación, se describe de forma abreviada el contenido de cada uno de los capítulos:

Capítulo 1- Fundamentación teórica sobre el Rehabilitador de Marcha

En este capítulo se analizan los conceptos de alto interés para la investigación como son los sistemas de RVi existentes en la actualidad, las principales características del RM, así como las herramientas y tecnologías necesarias para desarrollar la solución correspondiente a este trabajo de diploma.

Capítulo 2- Descripción y diseño de la solución

En el capítulo 2 se definen todos los elementos necesarios para la implementación del sistema que da respuesta al problema de la presente investigación. Se describen los módulos en que se divide la aplicación resultante y se realiza el diseño del sistema mediante los artefactos que indica la metodología de desarrollo seleccionada.

Capítulo 3- Implementación y validación de la solución

En este capítulo se materializan los requisitos identificados en el capítulo 2, implementándose el componente para incluir entornos virtuales y reajustar los parámetros de la marcha en relación con la posición de objetos colocados en las escenas para el RM. Se muestran los artefactos correspondientes a esta etapa según la metodología seleccionada, así como los diferentes estándares de codificación que se aplican durante la programación del código. También se muestran los resultados de las pruebas realizadas para comprobar el cumplimiento de los requisitos y la calidad del sistema.

Capítulo 1: Fundamentación Teórica sobre El Rehabilitador de Marcha

Con el fin de solucionar el problema planteado, se decide realizar un análisis de los aspectos fundamentales relacionados con el RM, para lo cual se seleccionaron los elementos de mayor influencia en el desarrollo de dicha solución. El presente capítulo se ha dividido en epígrafes según estos elementos esenciales. Inicialmente se analizarán las aplicaciones de RVi que existen en el mundo actualmente, luego el uso de entornos virtuales en estas y finalmente se describen las características principales del sistema RM, además del estudio realizado sobre las tecnologías y herramientas más adecuadas para el desarrollo de la solución.

1.1. Sistemas de Rehabilitación Virtual.

Actualmente existe una gran variedad de sistemas para la rehabilitación virtual que cuentan con un contexto funcional, concreto y estimulante para los pacientes, trayendo un beneficio directo tanto a estos como a los terapeutas por la adaptabilidad que tienen. Se emplean tecnologías de punta para la producción de ambientes simulados interactivos y multidimensionales; y mediante dispositivos visuales como monitores y lentes, dispositivos hápticos, entre otros, se logra sumergir al paciente en un entorno virtual y dotarlo de la capacidad de modificarlo en función de metas a cumplir. A continuación se describen algunos de estos sistemas, su desempeño y los logros alcanzados.

InTrainer es uno de los tantos ejemplos que existen actualmente de la aplicación de la RV en la medicina. Consiste en una solución a nivel de hardware y software, que permite a los pacientes ingresar en un mundo virtual para realizar sus tareas de entrenamiento físico, mientras son monitoreados por un sistema de adquisición de información biomédica, que reacciona y actúa dependiendo de los datos de entrada. Este proyecto se implementó en el área de rehabilitación cardiaca de un hospital y usa herramientas tecnológicas con el objetivo de captar la atención de los pacientes y centrarlos en el desarrollo de metas de corto plazo. El sistema obtiene información mediante sensores para procesarla y modificar el ambiente de forma interactiva, en dependencia de la situación actual de cada paciente (1).

En el año 2013 se desarrolló un marco de trabajo basado en RV para la rehabilitación física de miembros superiores en pacientes entre 6 y 12 años de edad. Está basado en una aplicación que utiliza videojuegos para la realización de actividades físicas particulares, captando la atención del individuo a través de su contenido dinámico y entretenido (2).

En ese mismo año también se empleó la RV para implementar un juego de mesa en un entorno simulado para la rehabilitación cognitiva (3).

Otro destacado ejemplo del empleo de entornos virtuales para facilitar la terapia de los pacientes lo constituye el Kaiten-zushi, sistema cuyo escenario virtual representa un sushi giratorio, en el cual los pacientes deben controlar los palillos virtuales mediante una interfaz háptica. Esta aplicación ha sido utilizada en diversos experimentos, en los que se han producido situaciones interesantes por la dinámica que se presenta. El sistema utiliza una serie de componentes para el desempeño de sus funciones como son: una pantalla CRT, un espejo, unas gafas de obturación, una cámara infrarroja que captura el seguimiento de la mirada, una interfaz háptica, un sistema de altavoces y dos ordenadores. Un ordenador gestiona el mundo virtual y el otro controla el sistema de seguimiento de la mirada (4).

El sistema de RVi llamado DEKA, fue llevado a cabo por la compañía de igual nombre con el propósito de ayudar a los usuarios a realizar y controlar diferentes movimientos. Facilita el aprendizaje motor mediante el uso de un entorno virtual, además de permitir el control de un avatar diseñado para manipular el brazo de Deka en el mundo real. Esta aplicación ha tenido resultados efectivos en personas que han padecido de amputaciones y que necesitan un ambiente de aprendizaje estructurado.

El sistema de brazo DEKA utiliza una serie de correas para controlar los movimientos del paciente, permitiendo que este emplee un conjunto de movimientos y active músculos que, por lo general, no coinciden con los utilizados para obtener la acción deseada en la rehabilitación tradicional (5).

De acuerdo con todos los sistemas analizados, el empleo de escenarios o entornos virtuales durante el proceso de rehabilitación de pacientes, permite la captación de la atención de los mismos y por consiguiente un mejor desempeño por parte de estos. Por tanto, se puede afirmar que la mayoría de los sistemas de RVi, que existen actualmente utilizan esta herramienta como medio para aumentar la efectividad del resultado de las terapias, acompañados de una serie de componentes que sirven de apoyo a los usuarios (ya sean pacientes o médicos), tales como: cámaras, computadoras, espejos, pantallas, correas, interfaces hápticas, etc. A continuación se realiza un análisis más profundo sobre el uso de ambientes o también denominados entornos virtuales dentro del mundo de la rehabilitación, así como las ventajas que proporciona.

1.2. Entornos Virtuales.

Un entorno virtual es el medio en el cual se realizan simulaciones de actividades que aparecen en la vida cotidiana, lo cual se realiza con el propósito de llevarlas a un ambiente controlado y analizarlas con mayor profundidad. Los medios virtuales de prueba permiten trabajar diferentes alteraciones de los mismos, llevando un estudio completo de la simulación deseada (6).

Una de las aplicaciones más recientes de la RV han sido los ambientes simulados para el tratamiento y evaluación de personas con discapacidades tanto intelectuales como físico-motoras. Son muchos los

inconvenientes para la realización de una terapia tradicional en pacientes con discapacidades de este tipo, por ejemplo la dificultad de lograr mediciones sobre el desempeño de una persona discapacitada cognitivamente. Por esta razón, los entornos virtuales se plantearon como una solución para disminuir los impedimentos de los métodos tradicionales.

La forma más común de aplicación de la RV desde el punto de vista de la neuropsicología ha sido en el ámbito del comportamiento adaptativo, en este sentido, los ambientes virtuales han sido empleados como métodos de entrenamiento de habilidades para la vida independiente y como formas de evaluación de la adquisición de tales habilidades. Un ejemplo de esto lo constituyen las investigaciones que utilizan el supermercado virtual para entrenar la habilidad de realizar compras, elegir artículos y ejecutar transacciones monetarias simples. El avance de los usuarios está dado debido a la similitud del ambiente virtual con un supermercado real, pero también a la simplicidad de su diseño que posibilita que el aprendizaje sea más flexible.

Los entornos virtuales son técnicas que han sido empleadas en disímiles ocasiones como es el caso del programa "ciudad virtual", arrojando resultados alentadores debido a que dichas tareas representativas de la vida real posibilitan a los usuarios con discapacidad intelectual aprender algunas habilidades básicas y entrenar las habilidades cognitivas superiores como la toma de decisiones, al menos en el corto tiempo que dura el entrenamiento (7).

Por parte de la rehabilitación física, estudios demuestran mediante pruebas ya realizadas con sistemas que emplean escenarios virtuales como juegos, en los que se deben evitar obstáculos o hacer tareas con rapidez que involucren los movimientos del tronco, que estos ayudan a fortalecer los mismos, dando una postura más erguida, mayor equilibrio y generando una base sólida para los movimientos de las extremidades. Estas evidencias prueban que existen muchas razones actualmente por las cuales incorporar el uso de ambientes virtuales en la rehabilitación de pacientes con discapacidades tanto físicas como cognitivas (8).

Razones por la cual usar entornos virtuales en la rehabilitación de pacientes con discapacidades físico-motoras y discapacidades cognitivas:

- Permiten simular aspectos de la vida cotidiana y ofrecer flexibilidad en el ajuste de parámetros: el entrenamiento intensivo, repetido, orientado a tareas, no siempre puede realizarse en el mundo real con objetos reales debido a las características clínicas de los sujetos. Con RV se pueden simular entornos y tareas motivantes y funcionales que en el mundo real el individuo no podría ejecutar con la misma precisión. Los sistemas de RVi permiten magnificar y adaptar la retroalimentación que el paciente recibe al realizar una tarea. Permiten asimismo programar la

intensidad y dificultad en función de los objetivos terapéuticos y las circunstancias individuales de cada usuario (9).

- La terapia resulta más emotiva y más interesante al paciente: permiten educar, entrenar y potenciar determinados aspectos de la aptitud física a través del juego y la competencia, grandes motivaciones. Esto posibilita trabajar de manera más efectiva con poblaciones a las que las terapias son difíciles de realizar, como niños o adultos mayores, debido a la posibilidad de empleo de juegos y escenarios más relajantes para los pacientes, distrayéndolos hasta cierto punto de su dolencia.
- Incremento de la tolerancia al dolor y el ejercicio: hay personas que debido a que el ejercicio les trae como consecuencia un incremento del dolor deciden no continuar la terapia. Se ha demostrado que aquellos que han probado programas de ejercicio utilizando RV se sumergen en un ambiente confortable. Se produce un efecto analgésico no farmacológico al distraerse, tolerando mucho mejor la terapia (8).
- Permiten trabajar condiciones de manera individual o global: se puede ser selectivo con los objetivos de una terapia.
- Menor riesgo: los ambientes controlados que se manejan con la realidad virtual son mucho más seguros y confiables.
- El hecho de que la RV le permite a las personas experimentar la sensación de control sobre sus procesos de rehabilitación, lo que resulta especialmente relevante dada la tendencia al comportamiento pasivo de estos individuos (10).

Todos los beneficios expuestos ponen de manifiesto que la utilización de ambientes virtuales puede ser una herramienta especialmente apropiada para personas con estos tipos de discapacidades. Seguidamente se realizará la descripción detallada del sistema de RVi Rehabilitador de Marcha, el cual constituye objetivo clave en este trabajo de diploma, para luego realizar la toma de decisiones correspondiente a qué elementos incluir y modificar en el software en desarrollo.

1.3. Rehabilitador de Marcha.

El sistema Rehabilitador de Marcha, entrenador de funciones motoras, se desarrolló en el año 2014 en la UCI en conjunto con el Centro de Rehabilitación Nacional Julio Díaz de La Habana, con el objetivo de poner al alcance de este país un sistema con grandes posibilidades para pacientes con discapacidades físico-motoras, esencialmente en las extremidades inferiores. El RM representa una alternativa cómoda

y segura para desarrollar la actividad de rehabilitación en pacientes que presenten dificultades severas al caminar. Con la utilización de esta herramienta se facilita el trabajo a los especialistas que llevan a cabo presencialmente la actividad de rehabilitación. Dicha aplicación emplea la RV para la realización de sus funciones.

El sistema posibilita al especialista gestionar los datos personales de un paciente y los relacionados con su padecimiento; de acuerdo a este parámetro el especialista podrá realizar en el sistema una serie de configuraciones para comenzar con una etapa previa de rehabilitación en el paciente. El sistema se adecuará a esos valores y proyectará los pasos de la marcha que debe seguir el paciente (11).

Todos estos valores que puedan ser configurados, tanto por el especialista como por el técnico que lleva a cabo las sesiones terapéuticas, podrán ser salvados y reutilizados para cualquier sesión terapéutica que sea similar con alguna otra. La aplicación permitirá crear un reporte que contendrá las diferentes configuraciones realizadas por el especialista para la realización de la marcha del paciente. Estas configuraciones serán separadas por sesiones para que el especialista en un primer momento pueda valorar el estado de avance del paciente (11).

Existen dos tipos de configuraciones dentro del RM: las configuraciones generales, donde se introducen los parámetros del tamaño del pie, de la orientación del pie y de las características de los pasos, mediante los cuales se especifica el ancho de los pasos y la longitud de los mismos y las configuraciones de proyección, que permiten modificar los parámetros de la cámara para la visualización de la marcha.

La aplicación, actualmente cuenta con una serie de menús que brindan diferentes opciones al médico especialista para manipular adecuadamente la marcha del paciente que se somete a la terapia. Ofrece la posibilidad de iniciar, pausar y detener la marcha, así como configurar los parámetros de proyección de la misma y además contiene otras funcionalidades para registrar los datos del paciente, configurar el sistema para cada uno de estos y guardar los valores de configuración como se ha descrito anteriormente.

La marcha se realiza de manera simple. Los pies del paciente se reflejan sobre la superficie de una estera o caminadora, sobre los cuales se realiza la marcha. La terapia se lleva a cabo de forma sencilla, como una caminata constante, que puede ser pausada y hasta invertida su dirección, pero no cuenta con ningún terreno de apoyo, ni con objetos que se interpongan en la marcha. Esto significa que la aplicación carece actualmente de entornos virtuales, de sonidos ambientes y de objetos que aparezcan en la escena, elementos fundamentales para el aumento de la efectividad del ejercicio.

Los entornos virtuales tienen una gran importancia en las aplicaciones de rehabilitación, debido a que estudios han demostrado que la inclusión de estos facilita el proceso y permite el logro de metas más altas en tiempos menores en los avances del paciente, así como hace menos dolorosas las secciones de rehabilitación o terapias. Es un método de distraer la mente humana hacia lo menos importante, para que lo más importante realmente, pase a un segundo plano y avance sin contratiempos bajo la dinámica del entretenimiento. Debido al gran número de ventajas que ofrece la inclusión de un ambiente virtual en la rehabilitación hoy en día, se decide en este trabajo de diploma agregar dicha funcionalidad al RM, al mismo tiempo que la inclusión de obstáculos en las escenas que contribuyan a aumentar el grado de dificultad del ejercicio para usuarios que lo requieran. A continuación, se describen los componentes fundamentales del RM, para luego determinar qué equipos o elementos son necesarios incorporar o modificar para la utilización del sistema en desarrollo.

1.4. Componentes del Sistema Rehabilitador de Marcha

El RM está compuesto hasta el momento por una caminadora eléctrica, con sus respectivas barras o brazos de seguridad a cada lado de la misma para el soporte y apoyo del usuario, un proyector de video en la parte frontal-inferior de la caminadora, para que proyecte en la estera la amplitud de los pasos, la cadencia, el ancho, así como el pie de cada usuario y el movimiento que debe realizar, y un ordenador encargado de tener el software que llevará a cabo todo este proceso y de ordenar la imagen que reflejará el proyector de video.

1.4.1. Caminadora eléctrica.

Una caminadora eléctrica, cinta de correr, cinta ergométrica, o máquina de caminar es un equipo con fines de entrenamiento físico, que puede funcionar mediante propulsión eléctrica o manual, la cual permite correr o caminar sin moverse del mismo sitio, usando una lámina de goma sintética que va cerrada como un lazo, denominada estera o tapiz. Su uso es sobre todo para la práctica de deporte, simulando el trote que se puede realizar en un espacio abierto. No obstante, también es utilizada para fines terapéuticos o de diagnóstico en centros médicos o de rehabilitación, como en el sistema del RM (12).

1.4.2. Proyector de video.

Un proyector de video o video proyector es un dispositivo óptico para capturar una imagen desde una fuente de video y proyectar la imagen correspondiente en una pantalla de proyección (en este caso el tapiz de la caminadora), usando un sistema de lentes, permitiendo así mostrar imágenes fijas o en movimiento. Todos los proyectores de video utilizan una luz muy brillante para proyectar la imagen, y los

más modernos pueden corregir curvas, borrones y otras inconsistencias a través de los ajustes manuales. La señal de video de entrada puede provenir de diferentes fuentes, en el RM está dado a través de una computadora personal (13).

Existen diferentes tipos de proyectores, el RM usa proyectores LCD. Suelen ser más baratos que otros, además de más eficientes en cuestiones de iluminación. Tienden a producir una imagen más nítida y no requieren de calibración ni de lugares especiales para su instalación.

1.4.3. Ordenador

Los ordenadores, sistemas de cómputo, o computadoras como suelen ser nombrados comúnmente son máquinas electrónicas que reciben y procesan datos para convertirlos en información conveniente y útil que posteriormente se envían a las unidades de salida (14).

Existen diversos tipos de ordenadores, el RM utiliza la computadora de escritorio, debido a que con sus prestaciones y características satisface las necesidades del sistema.

Describiendo los componentes que conforman el sistema RM en su totalidad, se puede concluir que, al adicionar entornos virtuales y sonidos ambientes, se necesitaría de un nuevo componente que permita visualizar dichos escenarios y reproducir dichos sonidos de modo tal que el paciente se sumerja en este entorno. Para darle solución a esta carencia se propone incorporar al sistema el uso de una pantalla plasma, colocándola en la parte frontal de la caminadora para que muestre tanto los pasos del usuario como el paisaje o terreno (entorno) por el cual transita y situar el proyector en el inferior de la pantalla.

1.4.4. Pantalla de plasma

Una pantalla de plasma es un dispositivo de pantalla plana habitualmente usada en la actualidad en televisores con un gran o pequeño formato. Consta de muchas celdas diminutas situadas entre dos paneles de cristal que contienen una mezcla de gases nobles (neón, argón y xenón). El gas en las celdas se convierte eléctricamente en plasma, el cual provoca que una sustancia fosforescente (que no es fósforo) emita luz potente. Tienen una amplia gama de colores y pueden fabricarse en tamaños bastante grandes, hasta 262 cm de diagonal. La principal ventaja de la tecnología del plasma es que pantallas muy grandes pueden ser fabricadas usando materiales extremadamente delgados, ya que cada píxel es iluminado individualmente (15).

Luego de haber realizado el análisis detallado sobre los aspectos relacionados con el sistema RM, sus características principales y las aplicaciones que realizan funciones similares, así como la obtención de los elementos necesarios para el desarrollo de la solución correspondiente a este trabajo de diploma, se

requiere definir las herramientas y tecnologías que se deben usar durante el proceso de desarrollo antes mencionado.

1.5. Metodologías y herramientas propuestas

1.5.1. Metodología de desarrollo

Existen dos grandes grupos de metodologías de desarrollo: las ligeras (ágiles) y las tradicionales (pesadas). Las primeras son las más apropiadas para llevar a cabo proyectos de poco volumen que requieran de una rápida implementación debido al corto período de tiempo disponible, mientras por su parte las segundas son apropiadas para proyectos grandes que por su importancia requieren una fuerte planificación y generación de documentación, por lo que se debería disponer de un tiempo más prolongado (16).

Las metodologías ágiles más utilizadas son SCRUM, Extreme Programming (XP) y Proceso Unificado Ágil (AUP por sus siglas en inglés), siendo esta última un híbrido de la metodología ágil XP y los artefactos de RUP. Mientras en otro orden las metodologías pesadas más frecuentemente utilizadas son Rational Unified Process (RUP) y Marco de Soluciones Microsoft (MSF).

Entre las metodologías de desarrollo antes descritas se decidió llevar a cabo este trabajo de diploma siguiendo una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado: XP. Para su selección se tuvo en cuenta los siguientes aspectos:

- XP es una metodología ágil que da lugar a una programación organizada, lo cual permite ahorrar en tiempo, aspecto fundamental para el desarrollo de esta solución, debido a que en este caso, el equipo de desarrollo es pequeño, compuesto solamente por una persona y dispone de un tiempo relativamente corto para la entrega de la primera versión funcional.
- XP se basa en la comunicación constante entre clientes y desarrolladores, la cual es una característica que se pone de manifiesto en este proceso de desarrollo, donde fluye un intercambio permanente de ideas entre el centro VERTEX y el programador.
- Esta metodología propicia además una pequeña tasa de errores, ya que fomenta la realización de pruebas continuas, lo cual facilita los cambios.
- XP es también, una metodología flexible, por lo que permite añadir o eliminar elementos de la documentación, según las necesidades del desarrollador, a medida que avanza el proyecto (17).

Una vez seleccionada la metodología que va a regir el proceso de desarrollo de la presente solución, es necesario elegir un lenguaje y una herramienta para llevar a cabo su modelado.

1.5.2. Lenguaje de Modelado

En todas las disciplinas de la ingeniería se hace evidente la importancia de los modelos ya que son quienes describen el aspecto y la conducta de los sistemas a desarrollar. El modelado es dividido a menudo en un número de vistas, cada una de las cuales describe un aspecto específico del producto o sistema en construcción. Es de gran importancia el modelado tanto para sistemas grandes como para pequeños, aunque mientras más grande sea el sistema más importante es el papel que juega el modelado en el mismo debido a que: *"El hombre hace modelos de sistemas complejos porque no puede entenderlos en su totalidad (18)."*

En el presente trabajo de diploma se ha decidido utilizar UML 2.0, ya que es una técnica para la especificación de sistemas en todas sus fases que cuenta con disímiles ventajas tales como una mejora de tiempos totales de desarrollo en un 50% o más, permite la facilidad de modelar sistemas utilizando conceptos orientados a objetos y establecer conceptos y artefactos ejecutables, además de brindar alta reutilización y minimización de costos. UML 2.0 da la facilidad de automatizar determinados procesos, aspecto de gran ayuda en este trabajo de diploma. No sólo por las ventajas que proporciona UML, se decide trabajar con este lenguaje, sino también debido a que el equipo de desarrollo ha experimentado otras experiencias con el mismo en proyectos anteriores (19).

1.5.3. Herramienta de Modelado

En la actualidad existen diversas herramientas de modelado como son ArgoUML (20), POSEIDON (21) y Visual Paradigm, las cuales utilizan el lenguaje de modelado UML.

En el caso de este trabajo de diploma se escogió Visual Paradigm en su versión 8.0 ya que es una herramienta considerada muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones (16). Su elección está dada principalmente por ser la herramienta usada en el RM, lo cual proporciona la reutilización de modelos, unido al soporte que proporciona al lenguaje de modelado seleccionado y a que genera documentación, brindando la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

1.5.4. Lenguaje de Programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente. Aunque muchas veces se usa lenguaje de programación y lenguaje informático como si fuesen sinónimos, no

tiene por qué ser así, ya que los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como, por ejemplo, el HTML (22).

Para el desarrollo del Sistema de Configuraciones Dinámicas y Ambientes Virtuales para el RM, se seleccionaron dos lenguajes de programación: C++: en su versión 98 para la aplicación desktop y C# en su versión 3 para la programación en Unity de los entornos virtuales. Se utiliza C++ puesto que es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel como la sobrecarga de operadores y la identificación de tipos en tiempo de ejecución, permite trabajar tanto a alto como a bajo nivel y la separación de un programa en módulos que admiten compilación independiente. Además posibilita la programación desde sistemas operativos, compiladores, aplicaciones de bases de datos, procesadores de texto o juegos (23). Por su parte C# presenta una gran sencillez de uso, debido a que elimina muchos elementos añadidos por otros lenguajes. Es un lenguaje moderno que incorpora elementos útiles que han demostrado a lo largo del tiempo ser beneficiosos para el programador. Es orientado a objetos como C++, por lo que soporta todos sus paradigmas como son polimorfismo, herencia y encapsulación. Además, es muy eficiente, todo el código incluye numerosas restricciones para garantizar su seguridad, no permitiendo el uso de punteros (24).

1.5.5. Marco de Trabajo

Para el desarrollo de la aplicación desktop correspondiente a este trabajo de diploma se ha decidido utilizar como marco de trabajo QT v.5.3, siendo este muy ventajoso debido a su libre adquisición a raíz de su condición de ser un software gratuito junto a todas sus herramientas. Además, es multi-plataforma, las aplicaciones desarrolladas en el mismo funcionan tanto en MAC y Windows algo difícil de lograr sin QT en la actualidad. También utiliza un editor visual, lo cual es de gran ayuda a la hora de ajustar dimensiones y características de algunos elementos en la interfaz de usuario y permite realizar interfaces más interesantes. Cuenta con una amplia documentación que lo respalda con más de 18 años, siendo esto último de gran utilidad pues sirve de guía y apoyo para desarrollar (25).

Por otro lado, para el desarrollo de los entornos virtuales tridimensionales con los cuales interactuará el usuario se ha decidido utilizar como marco de trabajo .NET en su versión 3.5 debido a que, al igual que QT, es un software libre, además de ser una plataforma orientada a objetos, brinda la facilidad de la interoperabilidad con código existente y presenta un fácil desarrollo basado en componentes. La plataforma también tiene una gran ventaja como lo es el soporte de bibliotecas de clases base (Base

Class Library), dichas clases proveen bloques básicos para construir aplicaciones, característica que facilita a gran escala el trabajo durante el proceso de desarrollo (26).

1.5.6. Entorno de Desarrollo

Un entorno de desarrollo debe ser elegido en dependencia de los lenguajes de programación y los marcos de desarrollo anteriormente seleccionados, el entorno que se va a utilizar para desarrollar la aplicación desktop para el RM es QTCreator versión 5.3, debido a que este se centra en proporcionar características que ayudan a los nuevos usuarios a aumentar la productividad. Brinda herramientas para la rápida navegación del código, resaltado de sintaxis y autocompletado del mismo, son características que describen algunas de sus ventajas, como también lo son: la ejecución línea por línea o instrucción por instrucción, los puntos de interrupción (breakpoints) y la interrupción de la ejecución del programa (27). Por otro lado, se utilizará Monodevelop 2.2 para el desarrollo de los entornos virtuales tridimensionales debido a sus numerosas ventajas como su particularidad de ser multiplataforma y además, presenta un ambiente amigable y simple. Proporciona una ayuda completa e incluye ejemplos de muchas soluciones. Posee autocompletado de sintaxis, importación de soluciones escritas con Microsoft, así como un navegador incorporado (28).

1.5.7. Motor gráfico y de desarrollo

En el proyecto se hace uso del motor gráfico orientado a objeto (OgreSDK_vc11_v1-9-0) pero no hace uso específicamente de las funcionalidades de OGRE solo que como está incluida en la arquitectura es necesario cargar los plugins o componentes concernientes a él para que de esta forma la arquitectura funcione.

Se seleccionó Unity 3D v5.3, que es un motor de desarrollo para la creación de juegos y contenidos 3D interactivos. Posee características como que es completamente integrado y que ofrece innumerables funcionalidades para facilitar el desarrollo de videojuegos, entre otros (29).

1.6. Conclusiones del Capítulo

Se caracterizó al sistema Rehabilitador de Marcha con el fin de identificar sus limitantes y se realizó un análisis detallado de los sistemas de RVi que existen en la actualidad, a partir de los cuales se definieron los elementos necesarios que deben estar presentes en la nueva aplicación a desarrollar.

Se determinó utilizar como metodología de desarrollo de software XP, como lenguaje de modelado UML 2.0 y la herramienta para llevar a cabo dicho modelado Visual Paradigm 8.0. También se decidió realizar la implementación de la solución en los lenguajes de programación C++ 98, para la aplicación de escritorio en la cual trabajará el especialista y C# 3 para el desarrollo de los entornos virtuales en los que se desempeñará el paciente. Se escogió el marco de desarrollo Qt en su versión 5.3 y para el desarrollo de los entornos virtuales .NET 3.5, además de los entornos de desarrollo (IDE) QtCreator versión 5.3 y Monodevelop 2.2 para una y otra respectivamente.

Capítulo 2: Descripción y Diseño de la Solución

Luego de haber realizado el análisis de los conceptos necesarios para la solución correspondiente a este trabajo de diploma, se describe detalladamente a continuación cada uno de los elementos que la componen. En este capítulo se representa también su diseño mediante los artefactos y los patrones de arquitectura que se requieren para la definición del sistema, sus funciones y su utilidad.

2.1. Descripción de la solución

El sistema Rehabilitador de Marcha, entrenador de funciones motoras, actualmente consta de tres módulos y brinda un total de 20 funcionalidades, las cuales en conjunto permiten al usuario médico especialista registrar los datos del paciente, administrar la marcha y configurar los parámetros de la misma según las necesidades de cada paciente, con el objetivo de lograr una terapia efectiva y exitosa (11).

La propuesta para la solución de esta investigación, llamada Sistema de Configuraciones Dinámicas y Ambientes Virtuales para el RM, consiste en un componente, que consta de dos módulos: Gestionar Obstáculo y Gestionar Entorno, los cuales se integran al sistema RM (como sub-módulos), formando parte del módulo Visualización. Todos en conjunto brindan al usuario la facilidad de controlar la proyección de la marcha de los pacientes. Este permite incorporar las opciones de insertar objetos y reajustar la marcha a partir de la posición de estos en la escena, incluir entornos tridimensionales donde se proyecten las imágenes de los pasos del paciente y añadir sonido para dichos entornos, lo cual le proporcionará al paciente un mejor proceso de rehabilitación en un ambiente más cómodo, con mayor eficacia en el resultado.

En la Fig. 1 se muestra un esquema, donde se representan todos los módulos del RM, incorporando los que forman parte de su sistema de configuraciones dinámicas y ambientes virtuales.

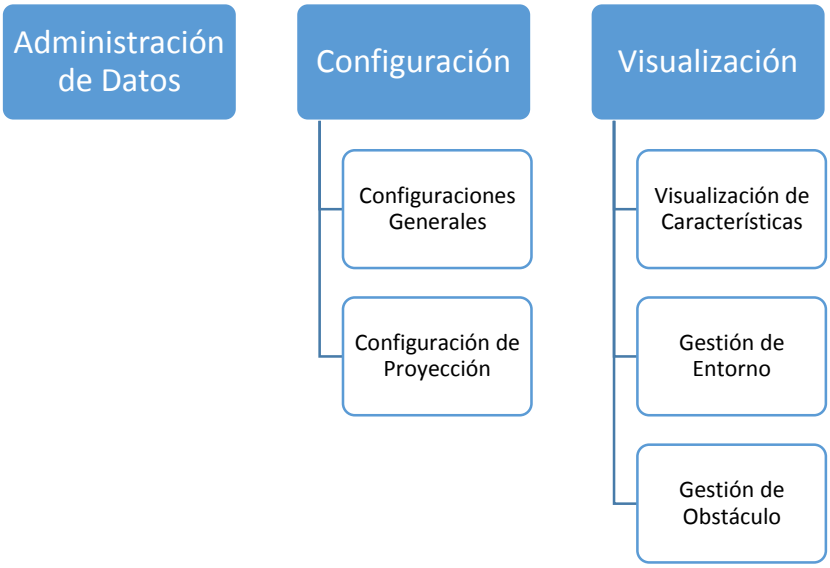

Fig. 1. Módulos del sistema.

Módulos de Administración de Datos y Configuración:

Estos módulos se encargan, de forma general, de la administración de los datos del paciente, permitiendo el registro de su información y la salva de sus configuraciones, así como la configuración de los pies, los pasos y la proyección de la marcha del paciente.

Visualización:

En principio, el módulo Visualización tiene como objetivo mostrar y ocultar el panel de Configuración que permite manipular los parámetros de los pies, los pasos y la proyección de la marcha del paciente, funcionalidades que se agrupan en el sub-módulo Visualización de características. Sin embargo, para darle respuesta a la necesidad que dio origen a la presente investigación, se propone integrar dos sub-módulos nuevos a este: Gestión de entorno y Gestión de obstáculo. Finalmente, dicho módulo estaría dividido en tres componentes, como se indica en el esquema anterior.

En el sub-módulo Gestión de Entorno se brinda al usuario la opción de seleccionar un entorno de los mostrados en la lista para realizar la marcha en este, el cual incluye un sonido ambiente. Además, permite cambiar de un entorno a otro u ocultar el seleccionado para realizar la marcha de manera simple.

En la Gestión de Obstáculo se agrupan todas las funcionalidades referentes al manejo del obstáculo, ya sea para insertarlo, eliminarlo o modificar sus parámetros. También incluye el reajuste de marcha que debe realizarse al insertar un obstáculo en esta.

La aplicación que se desea desarrollar debe contar con una interfaz de usuario que le permita al médico especialista insertar los datos de cada paciente en el sistema, para luego configurar según las características de cada individuo los elementos necesarios (tamaño del pie, orientación del pie, velocidad, ancho y longitud de los pasos, así como proyección de la cámara para la visualización de los pasos sobre el tapiz de la caminadora) para poner en función la marcha. Se propone que la aplicación ofrezca la posibilidad de realizar la marcha de dos formas: la forma clásica, ilustrándose los pasos sobre la estera o, en la que se selecciona la opción de mostrar entorno, para realizar el ejercicio en uno de los escenarios 3D de la preferencia del usuario, entre los soportados en el software. Esta última se realizará por medio de una pantalla de plasma y contendrá tres tipos de entornos virtuales de diferentes terrenos simulados, para que el paciente pueda sentirse en un ambiente más cómodo durante la terapia.

El Sistema de configuraciones dinámicas y ambientes virtuales para el RM permitirá cambiar de un entorno a otro, durante la realización del ejercicio. Cada entorno incluirá su propio sonido ambiente, con el fin de acercarse lo más posible a la realidad. En cada uno de ellos, se va a visualizar los pasos del paciente previamente configurados y la ejecución de la marcha, de modo que el individuo que se someta a la terapia se sumerja en este mundo virtual y logre con mayor efectividad el objetivo de la actividad. Además, se podrá realizar una marcha continua sin cambios o se podrá insertar objetos en medio de las escenas virtuales, si se desea aumentar el grado de dificultad del ejercicio, para escalar a un nivel más alto en el proceso de la terapia. La aparición de obstáculos en las escenas posibilita sacar al paciente de su marcha rutinaria, ya que la nueva aplicación en desarrollo brindará la funcionalidad de reajustar la marcha según la posición de los objetos que se visualizan en el terreno. También contará con la opción de que el médico especialista modifique las dimensiones del obstáculo a la misma medida que se realiza la rehabilitación.

La propuesta de solución de este trabajo de diploma contempla todas las opciones que brinda el RM ya puesto en funcionamiento, lo cual significa que las nuevas funcionalidades a añadir, no interfieren en la realización de las tareas básicas del sistema, contrario a esto, se ha pensado en una solución que permita realizar todas las funciones necesarias en conjunto con el empleo de ambientes virtuales y de obstáculos en las escenas. De este modo el nuevo sistema permitirá también la salva de configuraciones, la realización de reportes y la administración de la marcha (iniciar, pausar y detener marcha) sobre los terrenos virtuales.

Una vez realizada la descripción de la solución, se hace indispensable modelar el negocio, así de esta forma precisar la necesidad que existe actualmente y facilitar la comprensión del diseño de la solución de este trabajo de diploma.

Se decide realizar el diagrama de dominio, a pesar de que la metodología XP no define el flujo de trabajo modelado del negocio, ya que una de las características notorias de XP es su flexibilidad, por tanto, se puede incorporar sin dificultad estos elementos. *"Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. No se trata de un conjunto de diagramas que describen clases software, u objetos software con responsabilidades* (30)*."* En la siguiente Fig. 2 se muestra el diagrama de dominio, donde se representa la interacción entre todos los conceptos presentes en el negocio.

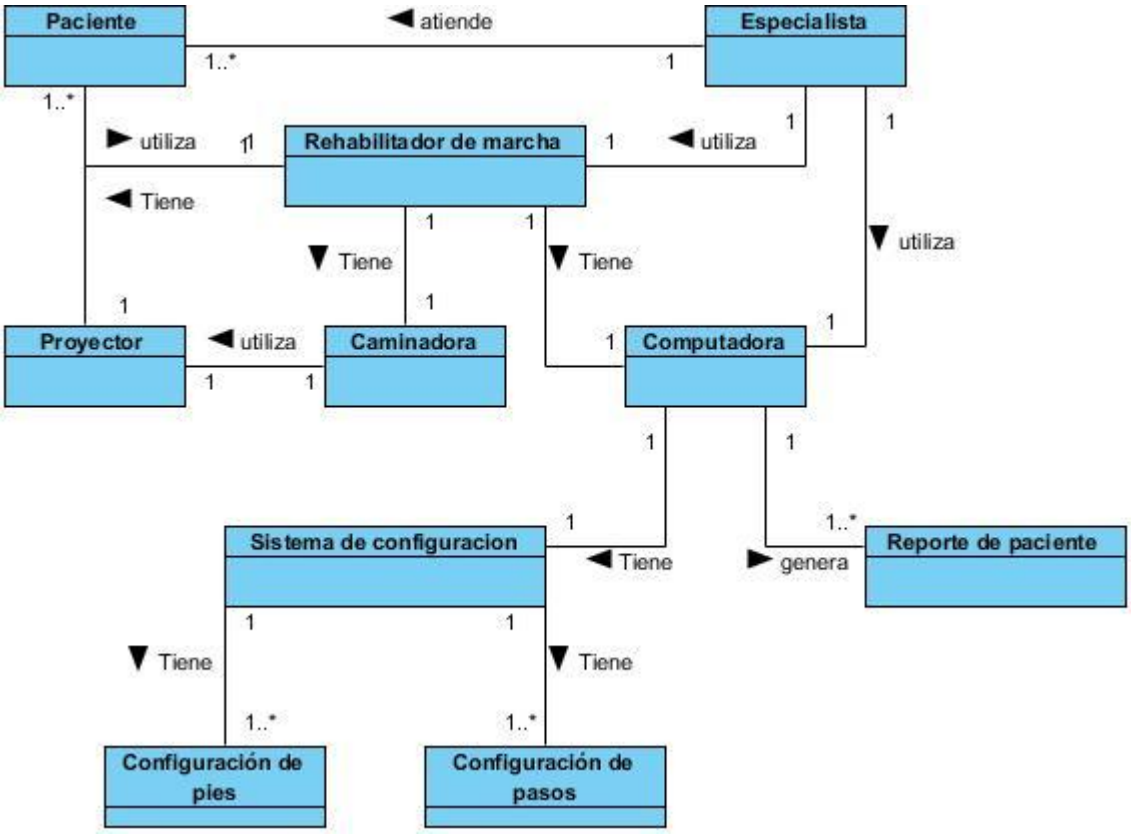

Fig. 2. Diagrama de Dominio

En la Fig. 2 se puede observar que, actualmente se cuenta con una aplicación llamada **Rehabilitador de Marcha**, la cual es utilizada por un médico **Especialista** para atender a un **Paciente**. Dicho **Rehabilitador de Marcha** cuenta con una **Caminadora**, un **Proyector** y una **Computadora**. La **Caminadora** utiliza el **Proyector** para realizar su función. La **Computadora** tiene un **Sistema de Configuración** y genera **Reportes de Pacientes**, al mismo tiempo que, el **Sistema de Configuración**

permite realizar **Configuraciones de Pies** y **Configuraciones de Pasos**, para poder modificar los parámetros de estos según las características de cada paciente.

2.1.1. Definición de las clases del modelo de dominio

Rehabilitador de Marcha: aplicación de escritorio desarrollada para la medicina, con el objetivo de ayudar a pacientes que necesitan rehabilitación en extremidades inferiores.

Especialista: médico especialista que está a cargo de la rehabilitación de los pacientes, usando el sistema RM. Es quien trabaja directamente con la computadora, configura los parámetros de los pies y los pasos de la marcha y genera los reportes de cada paciente.

Paciente: persona que presenta deficiencias físico-motoras en las extremidades inferiores y se inicia en el proceso de rehabilitación usando el RM.

Caminadora: forma parte del método de rehabilitación que se usa con los pacientes. Herramienta en donde se reflejan los pies que debe usar el paciente para realizar la marcha. Es un elemento clave para el progreso de la persona que lo utiliza.

Proyector: es el equipo que permite reflejar en la caminadora los pies del paciente con sus parámetros configurados para que este pueda iniciar la marcha.

Computadora: otra herramienta más usada en el método de rehabilitación del paciente. Es la encargada de proporcionarle la manipulación del sistema RM al especialista para que pueda trabajar con el paciente.

Sistema de Configuración: es el sistema contenido dentro de la aplicación RM, el cual brinda al usuario especialista las opciones de configurar los parámetros que poseen los pies y los pasos del paciente para la marcha.

Configuración de Pies: serie de parámetros de los pies del paciente que pueden ser configurados, como son ancho, largo y rotación.

Configuración de Pasos: permite configurar parámetros de los pasos del paciente, como son ancho, longitud y velocidad.

Reporte de Paciente: son reportes que se generan a través de la aplicación RM con los datos del paciente, lo cual le permite al médico especialista realizar diagnósticos sobre el progreso del mismo.

Luego de haber analizado la propuesta de la solución y los módulos que contendrá esta, se procede al diseño de la aplicación, el cual comprende desde la especificación de sus requisitos, tanto funcionales como no funcionales hasta la confección de los artefactos correspondientes a la metodología XP y la definición de sus clases.

2.2. Diseño de la aplicación

En el diseño de esta aplicación se tienen en cuenta los artefactos que define XP para la especificación de los requisitos del sistema, la planificación de las iteraciones a realizar y la representación de las clases presentes en la implementación. A continuación, se describe cada uno de estos artefactos.

2.2.1. Historias de usuario

Las historias de usuario permiten especificar tanto los requisitos funcionales como los no funcionales de una aplicación. Estas son elaboradas por el cliente en tarjetas de papel, usando un lenguaje común. Deben redactarse lo más explícitamente posible, de modo que se puedan extraer con facilidad los requisitos. Ofrece ventajas como son:

- Necesitan muy poco mantenimiento.
- Son adecuadas para proyectos donde los requerimientos son volátiles o poco conocidos.
- Al ser muy cortas, representan pequeños trozos de valor de negocio que se pueden implementar en un período de días o semanas.
- Mantienen un estrecho contacto con el cliente (31).

Durante el proceso de especificación de requisitos se identificaron 15 historias de usuario, las cuales se pueden consultar en la Planilla Historias de Usuario dentro del expediente de proyecto, adjunto a este trabajo de diploma. De estas, se extrajeron 15 requisitos funcionales y 17 no funcionales, los cuales se exponen más adelante. En la Tabla 1 se observa una de las historias de usuario con mayor relevancia para la solución. Describe la funcionalidad Configurar parámetros de obstáculo.

Tabla 1. Historia de Usuario Configurar parámetros de obstáculo.

Historia de Usuario	
Número: 5	**Nombre de la Historia de Usuario:** Configurar parámetros de obstáculo.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Especialista	**Iteración asignada:** 1
Prioridad en el negocio: Muy Alta	**Puntos estimados:** 1.4 semanas
Riesgo en desarrollo: Alto	**Puntos Reales:** 1.4 semanas
Descripción: La aplicación debe permitir al especialista modificar las dimensiones del obstáculo a la misma medida que se realiza la rehabilitación.	

Observaciones: Para poder realizar esta operación se debe haber seleccionado previamente un entorno virtual.
Prototipo de Interfaces: 

2.2.2. Lista de reserva del producto

Otro de los artefactos que define la metodología XP es la lista de reserva del producto. Su objetivo es listar todos los requisitos que debe cumplir la aplicación, ordenando los funcionales por prioridad desde muy alta hasta prioridad baja, junto a la estimación del tiempo en semanas que va a demorar la implementación de cada una de estas funcionalidades y el rol de la persona que realiza la estimación. Luego se definen los requisitos no funcionales separados por categoría, dígase de Seguridad, de Software, de Hardware, etc. A partir de los requisitos identificados se elaboró la Lista de Reserva del Producto de la solución para el Sistema de Configuraciones Dinámicas y Ambientes Virtuales para el RM, la cual se muestra en la

Tabla 2 que aparece a continuación.

Tabla 2. Lista de Reserva del Producto

Item *	**Descripción**	**Estimación**	**Estimado por**
Prioridad: Muy Alta			

1	Reajustar marcha	1.6 semanas	Analista
2	Visualizar secuencia de pasos de la marcha en el entorno virtual	1.4 semanas	Analista
3	Mostrar obstáculo	1.4 semanas	Analista
4	Configurar parámetros de obstáculo	1.4 semanas	Analista
5	Iniciar marcha en el entorno virtual	1 semana	Analista
6	Modificar entorno virtual	1 semana	Analista
Prioridad: Alta			
7	Seleccionar entorno virtual	0.4 semanas	Analista
8	Ocultar obstáculo	0.6 semanas	Analista
9	Ocultar entorno virtual	0.6 semanas	Analista
10	Pausar marcha en el entorno virtual	0.8 semanas	Analista
11	Detener marcha en el entorno virtual	0.6 semanas	Analista
Prioridad: Media			
12	Visualizar en el reporte los parámetros de los obstáculos	0.4 semanas	Analista
13	Visualizar en el reporte el entorno.	0.4 semanas	Analista
Prioridad: Baja			
14	Visualizar en el reporte la utilidad de los obstáculos.	0.2 semanas	Analista
15	Visualizar en el reporte la utilidad de los entornos.	0.2 semanas	Analista
Requisitos No Funcionales			
Usabilidad			
1	El sistema a desarrollar posee un dominio de aplicación de escritorio.		

2	El sistema es muy sencillo de usar, ya que comprende una fácil navegación por los diferentes contenidos que incluye, teniendo como premisa fundamental la correcta estructuración y organización de la información que establece la disciplina de arquitectura de información. Esto se tiene en cuenta para que el usuario tenga el conocimiento y el dominio requerido del lugar del sistema en el que se encuentra.		
Confiabilidad			
3	Ante cualquier evento adverso, ya sea correcto o incorrecto para darle cierta seguridad al sistema se lanzará una excepción en caso de alguna advertencia.		
Eficiencia			
4	El tiempo de respuesta del sistema ante cualquier acción que realice el usuario será de 1 segundo.		
5	La aceleración gráfica para visualizar la marcha deberá incluir los controladores propios que deberán estar instalados dependiendo del hardware gráfico.		
Restricciones de Diseño e Implementación			
6	Lenguaje de programación a utilizar C++ bajo el IDE QT Creator 5.3.		
7	La biblioteca de desarrollo utilizada es Qt en su versión 5.3.		
Requisitos para la documentación de usuarios en línea y ayuda del sistema			
8	El sistema deberá mostrar una ayuda inmediata para cualquier operación que vaya a realizar el usuario en el sistema. Para ello la aplicación permitirá seleccionar en cualquier momento el sistema de ayuda para el usuario.		

Software			
9	El sistema deberá ejecutarse sobre sistema operativo superior a Windows 7.		
Hardware			
10	El sistema contará para ejecutarse con 2 entradas de tarjeta de video independientemente del modelo que se pueda utilizar, ya que el sistema no requiere específicamente de muchos gráficos.		
11	La capacidad requerida de la memoria RAM deberá ser de igual o mayor a 1 Gigabyte.		
12	El disco duro que puede soportar la aplicación será mayor o igual a 80 Gigabyte.		
Interfaz Gráfica o Apariencia Externa			
13	Las interfaces estarán sustentadas por los colores comunes de presentación que contiene cualquier formulario que son el gris, blanco y negro.		
Requisitos de Licencia			
14	Licencia GPL: QT Creator versión 5.3 (IDE de desarrollo).		
15	Licencia GPL: Framework Qt versión 5.3 (IDE de desarrollo).		
Estándares Aplicables			
16	Los estándares de calidad utilizados durante el desarrollo de la aplicación serán los establecidos por el Centro Nacional de Calidad de Software (Calisoft). Los mismos serán conciliados entre las partes durante la preparación de las pruebas.		

17	Otros estándares que se tuvieron en cuenta en el desarrollo del software fueron los relacionados con el código de implementación a utilizar que se incluyen en documento Estándar_Codificación.pdf que definió el centro VERTEX.		

2.2.3. Plan de Iteraciones

El plan de iteraciones permite organizar el proceso de desarrollo de la aplicación ya que mediante él se puede planificar el tiempo que se demora en implementar todas las funcionalidades necesarias para llevar a cabo una iteración completa. En este artefacto también se define la prioridad en el negocio para las funcionalidades ya que de esto depende las iteraciones que se van a desarrollar. En la Tabla 3 se representa el Plan de Iteraciones confeccionado para la aplicación que se corresponde con el presente trabajo de diploma.

Tabla 3. Plan de Iteraciones

Número	**Descripción de la Iteración**	**Historias a Implementar**	**Duración Total**
1	Se desarrollan las historias de usuario más importantes para el negocio, relacionadas con la gestión de entornos virtuales y obstáculos en la marcha.	11, 1, 3, 5, 9, 7	7,8 semanas
2	Se desarrollan las funcionalidades de detención de la marcha en el entorno virtual, así como las que permiten ocultar obstáculos y entornos.	2, 4, 6, 10, 8	3 semanas
3	En esta iteración se agrupan las funcionalidades que permiten la visualización en el reporte de los parámetros de los obstáculos y del entorno.	14, 15	0.8 semanas
4	En esta iteración se agrupan las funcionalidades que permiten la visualización en el reporte de la utilidad de los obstáculos y de los entornos.	12, 13	0.4 semanas

2.2.4. Tarjetas CRC

Estas tarjetas se dividen en tres secciones, las cuales contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores, de ahí proviene su nombre Clase-Responsabilidad-Colaboración (CRC). Una clase es cualquier persona, evento, objeto, concepto, pantalla o reporte. Las responsabilidades de una clase es todo aquello que conoce y que realiza, sus atributos y métodos, mientras que los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades. Resulta muy útil tener pequeñas tarjetas de cartón para llenarlas y mostrárselas al cliente con el objetivo de llegar a un acuerdo sobre la validez de las abstracciones propuestas (32). Para el desarrollo del sistema se debe elaborar una tarjeta por cada clase presente y en este caso se identificaron un total de 10 tarjetas CRC. La Tabla 4 que se muestra a continuación, representa la Tarjeta CRC para la clase ObstacleDeployer, mientras que el resto de las tarjetas se pueden encontrar en el expediente de proyecto adjunto a este trabajo de diploma.

Tabla 4. Tarjeta CRC de la clase ObstacleDeployer

Tarjeta CRC	
Clase: ObstacleDeployer	
Responsabilidades	**Colaboraciones**
-Configurar el largo del obstáculo. -Determinar la frecuencia con que aparece el obstáculo.	SteraManager

Una vez concluido los artefactos que documentan el Sistema de Configuraciones Dinámicas y Ambientes Virtuales para el RM, se hace necesario representar los elementos claves del diseño y la arquitectura de este, lo cual se realiza mediante el diagrama de clases, que permite mostrar con mayor claridad la información descrita en las tarjetas CRC.

2.2.5. Diagrama de Clases

La metodología XP no define el diagrama de clases, pero se ha decidido añadir este elemento al trabajo de diploma con el objetivo de representar con más detalle el diseño del sistema y así poder visualizar las relaciones entre las clases que se involucran en él.

"El propósito de un diagrama de clases es describir las clases que conforman el modelo de un determinado sistema. El diagrama de clases va a ser creado y refinado durante las fases de análisis y diseño, estando presente como guía en la implementación del sistema (33)*."*

En este trabajo de diploma se representaron dos diagramas de clases, uno se corresponde con la versión inicial del RM y las nuevas clases que le integra la presente solución, mientras el otro diagrama modela las clases del nuevo componente que se le inserta al sistema. A continuación en la Fig.3 y la Fig. 4 se puede observar con detalle las clases que conforman dichos diagramas.

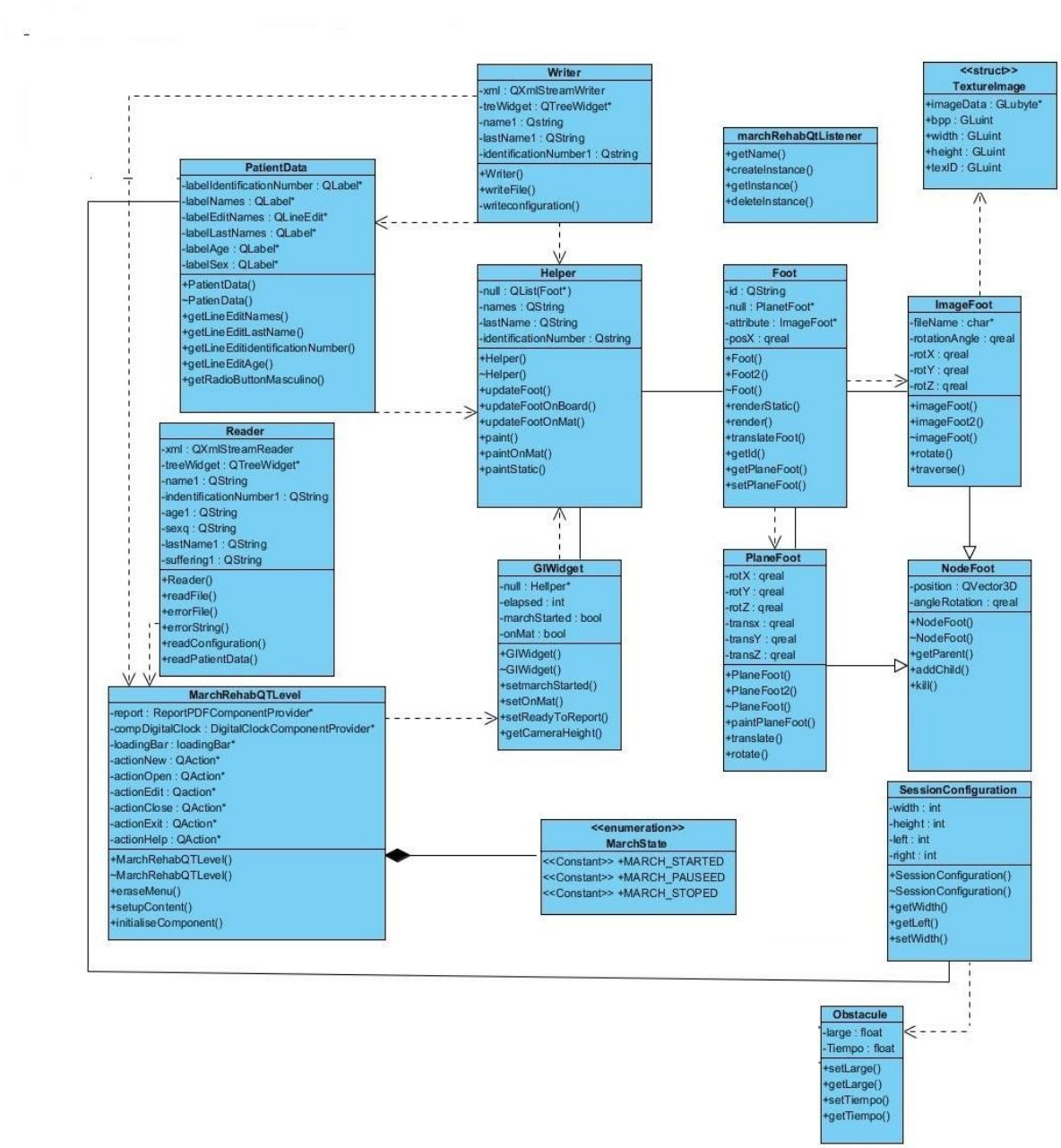

Fig.3.Diagrama de clases de aplicación de escritorio

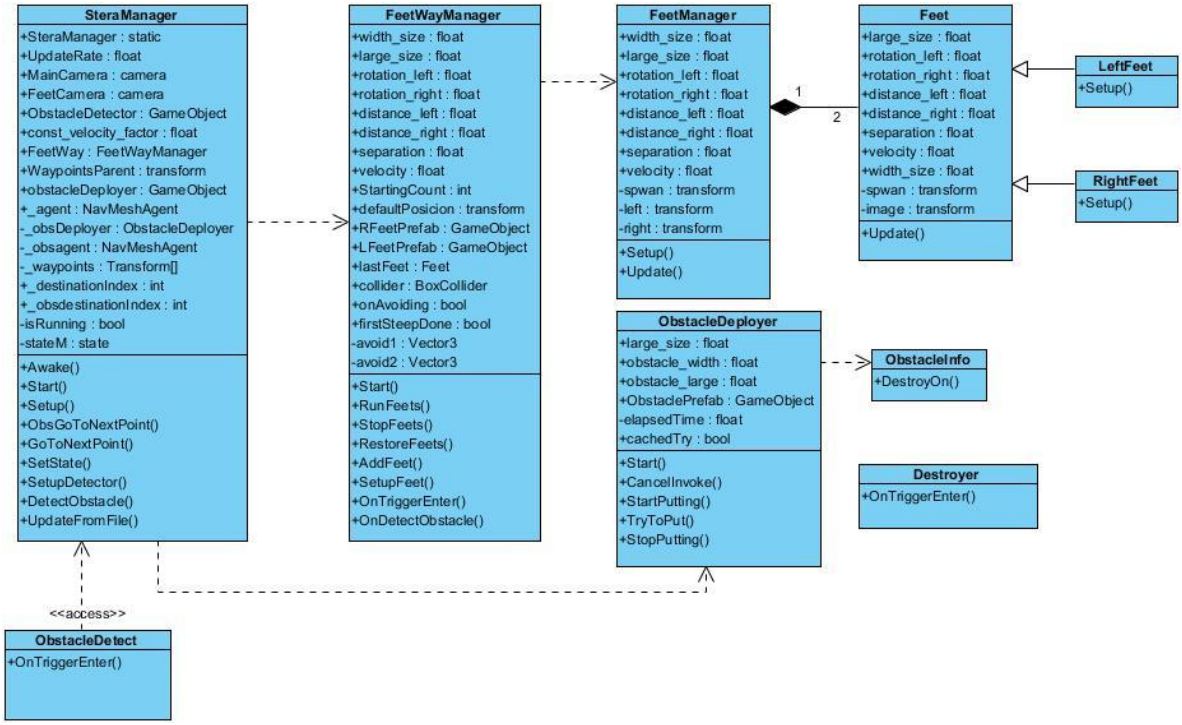

Fig. 4. Diagrama de clases de entornos virtuales

2.3 Arquitectura base de la aplicación

La arquitectura del software es el diseño de más alto nivel de la estructura de un sistema, esta define de manera abstracta los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Es de gran importancia debido a que proporciona una excelente vista del proceso de desarrollo de la aplicación, así como una mejor comprensión del sistema para todos los que vayan a intervenir en él (34). La arquitectura que se llevará a cabo en el Sistema de Configuraciones Dinámicas y Ambientes Virtuales para el RM, está constituida por estilos y patrones de arquitectura de diseño que a continuación se describen.

2.3.1 Estilos y patrones arquitectónicos

Los patrones y estilos arquitectónicos son un tema abordado por muchos autores de relevante importancia que han dedicado gran parte de su vida al estudio de dicha materia. En la documentación consultada se observó una mayor inclinación por el uso de forma indiferente de ambos términos, donde se deja ver de una forma concisa que las diferencias entre estos radican principalmente en el nivel de abstracción. Incluso los nombres de algunos estilos y patrones arquitectónicos coinciden en muchos textos, sobre este tema aborda Carlos Reynoso.

"Existen claras convergencias entre ambos conceptos, aun cuando se reconoce que los patrones se refieren más bien a prácticas de re-utilización y los estilos conciernen a teorías sobre las estructuras de los sistemas a veces más formales que concretas. Algunas formulaciones que describen patrones pueden leerse como si se refirieran a estilos, y también viceversa. En cuanto a los patrones de arquitectura, su relación con los estilos arquitectónicos es perceptible, pero indirecta y variable incluso dentro de la obra de un mismo autor (35)."

En este caso, la arquitectura de software del sistema está regida y llevada a cabo por la empleada en los Laboratorios Virtuales, usando por ende el mismo patrón arquitectónico definido en la tesis de grado titulada Laboratorios Virtuales del autor Alejandro David Merzeau Martínez. Es una arquitectura basada en componentes que facilita el trabajo fluido y la interacción de varios modelos con otros. Un componente es una unidad de composición de aplicaciones, que posee un conjunto de interfaces especificadas contractualmente y dependencias del contexto explícitas, que ha de ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio (36). Además, la arquitectura utilizada en el sistema en cuestión, agiliza la forma de agregar nuevas funcionalidades por medio de dichos componentes, siendo esto uno de los objetivos principales de este trabajo. Gracias a estas facilidades que brinda este método de trabajo se puede contar con componentes de una fácil manipulación e inserción en el sistema como una buena práctica de reutilización de los mismos, tales como Component_DigitalClock encargado de manejar el tiempo de realización del ejercicio, Component_ReportPDF encargado de la generación de los reportes en formato PDF y License_manager para gestionar las licencias del software.

2.3.2 Patrones de diseño

Los patrones de diseño generalmente se utilizan ya que permiten la programación de clases reutilizables y la reutilización de clases ya existentes. Es aconsejable el uso de estos patrones durante la realización del diseño de un software ya que son soluciones simples que indican cómo resolver un problema particular utilizando un pequeño número de clases relacionadas de forma determinada (37).

2.3.3 Patrones GRASP

Seguidamente se expondrán los patrones de asignación de responsabilidades (GRASP por sus siglas en inglés) que se emplean en la arquitectura del sistema en desarrollo, los cuales son indispensables para la correcta programación del código posteriormente.

Patrón Experto

Experto en información es el principio básico de asignación de responsabilidades, o sea este indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la

clase que conoce la información necesaria para crearlo (30). En la Fig. 5 se muestra cómo la clase SteraManager le asigna la responsabilidad a la clase ObstacleDeployer de gestionar la información de ObstacleInfo.

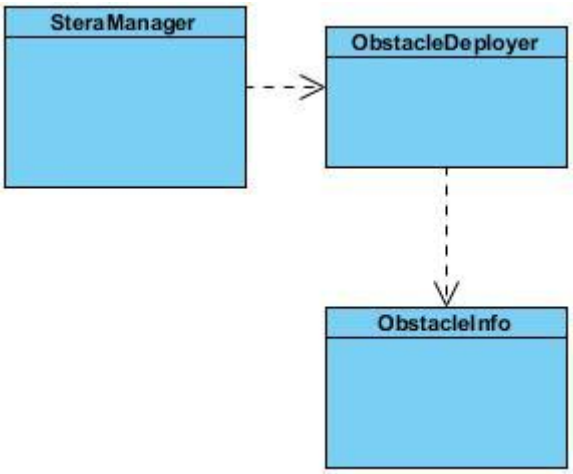

Fig. 5. Evidencia del patrón Experto

Patrón Creador

Este patrón se puede identificar cuando una clase tiene la responsabilidad de crear una instancia de otra clase, lo cual se puede dar en diferentes casos:

- B agrega los objetos A
- B contiene los objetos A
- B registra las instancias de los objetos A
- B utiliza específicamente los objetos A
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (30).

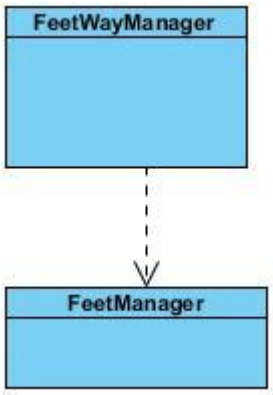

Fig. 6. Evidencia del patrón Creador

Una de las consecuencias de utilizar dicho patrón es la visibilidad entre la clase creada y la clase creador, en la Fig. 6 se muestra como la clase FeetManager utiliza los objetos de FeetWayManager.

Patrón Alta Cohesión

Consiste en asignar una responsabilidad de modo que la cohesión siga siendo alta. Una clase que tenga alta cohesión posee un número relativamente pequeño, con una importante funcionalidad relacionada y poco trabajo por hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande (30). En la Fig. 7 se representa el empleo de este patrón ya que ambas clases trabajan en conjunto para que se lleven a cabo las tareas de la clase ObstacleInfo.

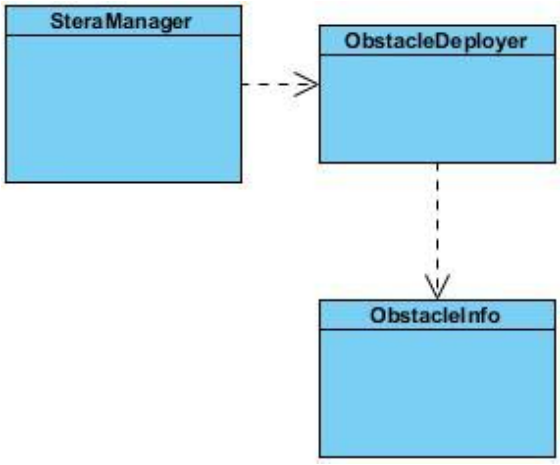

Fig. 7. Evidencia del patrón Alta Cohesión

Patrón Bajo Acoplamiento

El patrón bajo acoplamiento defiende la idea de tener las clases lo menos ligadas posibles entre sí, de tal forma que en caso de producirse alguna modificación o error en alguna de ellas, repercuta de la menor manera posible en el resto de las clases (30). En la Fig. 8 sale a relucir el bajo acoplamiento de estas clases ya que FeetManager no depende de muchas otras clases, solo de FeetWayManager.

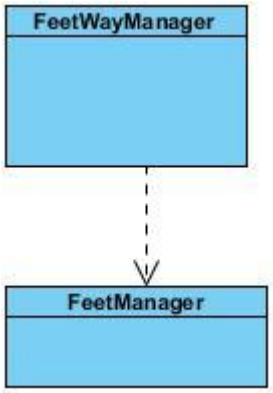

Fig. 8. Evidencia del patrón Bajo Acoplamiento.

2.3.4 Patrones GoF

Cuando se menciona el tema de patrones de diseño resulta necesario tomar en cuenta el libro “Design Patterns: Elements of Reusable Object Oriented Software”. Este influyente material fue escrito por Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides, también conocidos como “Gang of Four” (GoF). En él se presenta un catálogo de 23 patrones divididos en 3 categorías: creacionales, estructurales y comportamiento. Los autores presentan los patrones de diseño a partir de sistemas reales. Es por esta razón que estos patrones no son elementos puramente teóricos, sino que están fundados sobre una fuerte base práctica. Un patrón de diseño es una descripción de clases y objetos que se comunican entre sí, adaptados para resolver un problema general de diseño en un contexto particular (38).

El patrón GoF aplicado en el diseño de la solución de esta investigación es:

Cadena de Responsabilidades: es un patrón de comportamiento que evita acoplar el emisor de una petición a su receptor dando a más de un objeto la posibilidad de responder a una petición. Para ello, se encadenan los receptores y pasa la petición a través de la cadena hasta que es procesada por algún objeto (39). Este patrón es utilizado a menudo en el contexto de las interfaces gráficas de usuario donde un objeto puede estar compuesto de varios objetos como muestra la Fig. 9. El marco de trabajo

seleccionado para el desarrollo del sistema, utiliza por defecto dicho patrón, por consiguiente, está presente en su diseño.

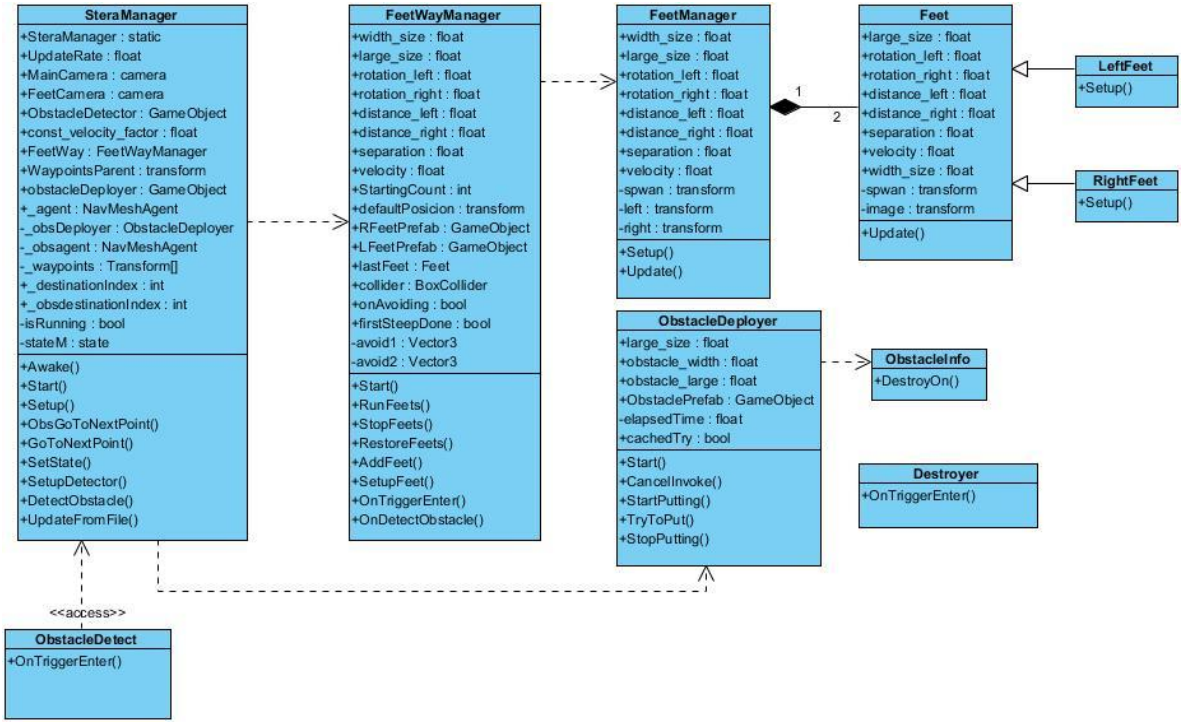

Fig. 9. Evidencia del patrón Cadena de Responsabilidades.

2.4 Conclusiones del Capítulo

En este capítulo se realizó el diseño del Sistema de Configuraciones Dinámicas y Ambientes Virtuales para el RM, en el cual se obtuvieron 15 historias de usuario, y de estas se especificaron 15 requisitos funcionales y 17 requisitos no funcionales para el sistema a desarrollar. Se conformó el plan de iteraciones, definiéndose 4 iteraciones en total para llevar a cabo el desarrollo de la solución. Además se representó el diagrama de clases y se diseñaron las tarjetas CRC para cada una de estas, obteniéndose 10 finalmente.

Se utilizó el patrón arquitectónico Basado en Componentes y como patrones de diseño, Bajo Acoplamiento, Alta Cohesión, Creador y Experto, los cuales se evidencian en el diagrama de clases representado.

Capítulo 3: Implementación y Validación de la Solución

La implementación es la etapa en la que se materializa el resultado principal de todo el flujo de trabajo, no es más que la obtención de componentes en los cuales existen ficheros, ejecutables y sus dependencias. Es el resultado visible a ojos del cliente, necesario e imprescindible para cumplir con sus expectativas. Luego de haber concluido el proceso de diseño de la aplicación se deben especificar las tareas de ingeniería que permiten organizar el proceso de desarrollo de las diferentes historias de usuario con el objetivo de llevar a cabo la implementación de una forma más sencilla.

1.1 Implementación de la solución

3.1.1. Tareas de Ingeniería

Las tareas de ingeniería son las encargadas de describir las pautas trazadas y que se llevan a cabo en el proyecto. Están clasificadas en diferentes tipos, de desarrollo, corrección, mejora, entre otras. Las tareas tienen relación con una historia de usuario; se especifica la fecha de inicio y fin de las mismas, se menciona al programador responsable de que sea cumplida y también se describe lo que será llevado a cabo en la tarea (40). Para la solución de este trabajo de diploma se definieron un total de 34 tareas de ingeniería, las cuales pueden consultarse en el expediente de proyecto adjunto a este trabajo de diploma. A continuación en la Tabla 5 se muestra una tarea de ingeniería para la historia de usuario “Reajustar Marcha”.

Tabla 5. Tarea de Ingeniería Implementar Reajustar marcha

Tarea de Ingeniería	
Número Tarea: 28	**Numero Historia de Usuario:** 11
Nombre Tarea: Implementar Reajustar Marcha	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1.0 semanas
Fecha Inicio: 6/03/2017	**Fecha Fin:** 13/03/2017
Programador Responsable: Jorge Raydel Sánchez Garrote	
Descripción: El reajuste de marcha está determinado según la amplitud del obstáculo. La captura de la posición del obstáculo en el espacio es determinada por un detector el cual se ubica delante de la cámara, este detector es un colisionador. Cuando colisiona dispara una señal y con esa señal él determina donde va el próximo paso, el paso anterior se calcula en base al espacio que queda y el posterior se calcula según la amplitud del obstáculo.	

Además de las tareas de ingeniería, otros elementos importantes para la implementación de una aplicación lo constituyen los estándares de codificación, tema que se analizará a continuación.

3.1.2. Estándares de codificación

Los estándares de codificación proporcionan una especie de lenguaje común para facilitar la implementación del código a todos los miembros de un equipo de desarrollo. Su empleo es de vital importancia para lograr un estilo y una arquitectura robusta. Además posibilita una mejor comprensión del código escrito por otro, lo cual resulta imprescindible para la facilidad de reutilización y mantenimiento (41).

En lo siguiente se exponen los estándares de codificación utilizados en la implementación del sistema.

- Uso de la convención Pascal para el nombre de las clases

La primera letra del identificador y la primera letra de las siguientes palabras concatenadas se escriben en mayúsculas. El estilo de mayúsculas y minúsculas Pascal se puede utilizar en identificadores de tres o más caracteres, como se pone de manifiesto en el siguiente código de la aplicación:

```
public class SteraManager: MonoBehaviour {
...
}
```

- Uso de la convención Pascal para el nombre de los métodos.

```
void Awake()
{
    Instance = this;
}
```

- Uso de la convención Camel para el nombre de variables y parámetros, donde la primera letra del identificador está en minúscula y la primera letra de las siguientes palabras concatenadas en mayúscula, como en el siguiente fragmento de código extraído de la aplicación de este trabajo de diploma:

```
public void SetupDetector(Vector3 fromPosition)
{
```

```
    Vector3 prefPosition = MainCamera.ViewportToWorldPoint(fromPosition);
    Transform detector = ObstacleDetector.transform;
    detector.position = prefPosition;
}
```

- Acerca de las llaves y líneas en blanco, se abren y cierran las llaves en los renglones siguientes a los que se encuentra el código del método, tal cual el ejemplo siguiente de la aplicación:

```
public void DetectObstacle(GameObject obstacle)
{
    FeetWay.OnDetectObstacle(obstacle);
}
```

- Indentación y espaciado

```
    public void SetState(State state)
    {
  stateM = state;
            switch (state)
            {
            case State.Started:
        if (!isRunning)
        {
            GoToNextPoint();
            ObsGoToNextPoint();
            isRunning = true;
            FeetWay.RunFeets();
        }
        else
```

```
        {
            __obsagent.Resume();
            __obsDeployer.StartPutting();
            __agent.Resume();
            FeetWay.RestoreFeets();
        }
        break;
            case State.Paused:
        if (isRunning)
        {
            __obsagent.Stop();
            __obsDeployer.StopPutting();
            __agent.Stop();
            FeetWay.StopFeets();
        }
        break;
            case State.Stopped:
        if (isRunning)
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
                    break;
            }
    }
```

3.1.3 Principales Interfaces de la Aplicación

Realizando las tareas de ingeniería definidas y siguiendo los estándares de codificación antes descritos se logró una adecuada programación del sistema. Seguidamente se exhiben el resultado final de algunas de las interfaces de la aplicación.

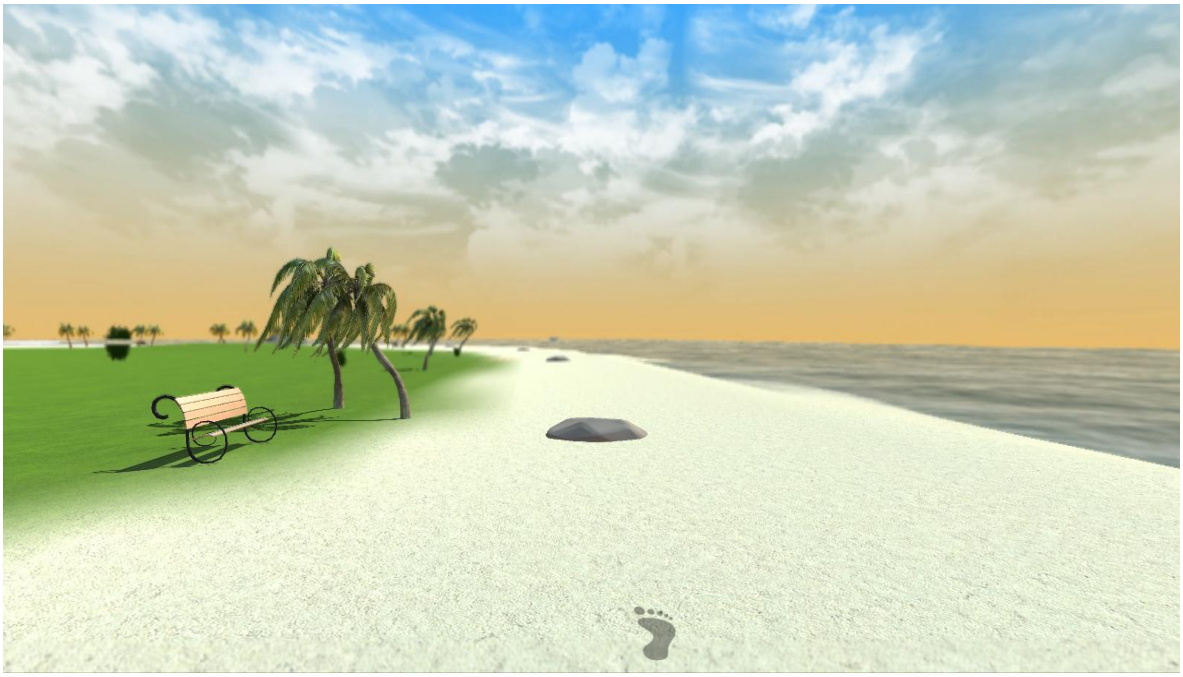

Fig. 10. Vista del Entorno Virtual Playa

Fig. 11. Vista del Entorno Virtual Bosque

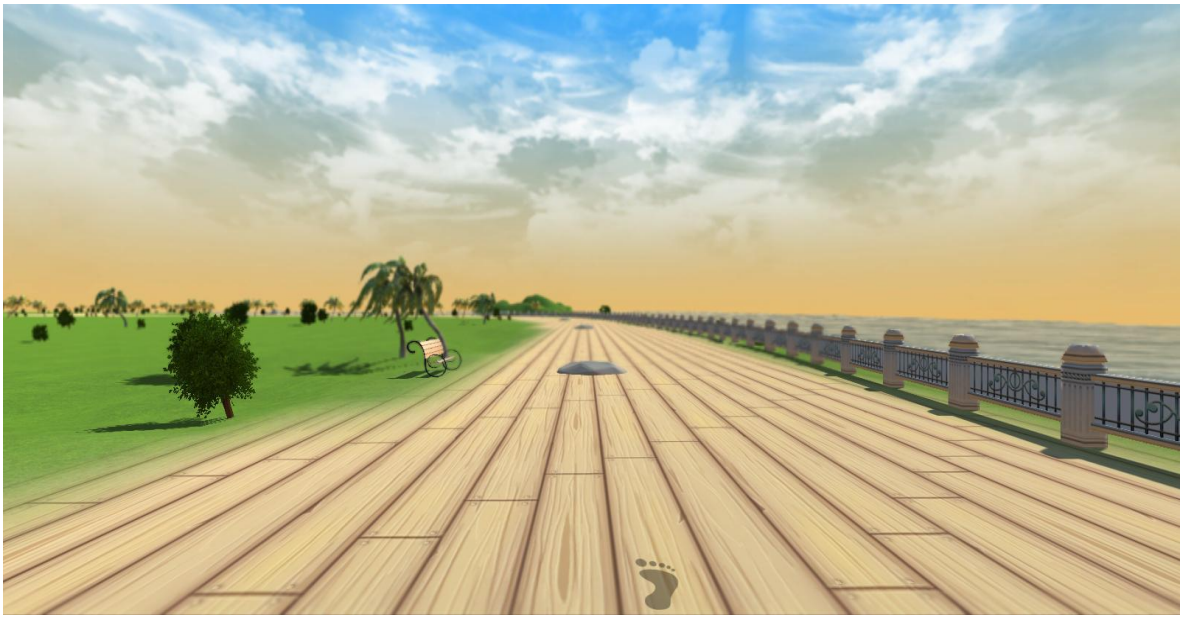

Fig. 12. Vista del Entorno Virtual Malecón

En la Fig. 10, Fig. 11 y Fig. 12 se pueden observar los entornos virtuales Playa, Bosque y Malecón en ese mismo orden, los cuales son los tres ambientes insertados en la aplicación y de relevante importancia, puesto que es por donde transitará el paciente durante su rehabilitación. Además, se pueden visualizar los obstáculos que van apareciendo en las escenas, punto importante que se tuvo en cuenta para el desarrollo del entorno virtual por la relevancia que trae consigo introducir el mismo a la rehabilitación desde el punto de vista médico.

Fig. 13. Vista Principal del Rehabilitador de Marcha

En la Fig. 13 se puede observar la interfaz principal de la aplicación, con la que más interactúa el especialista. La misma está constituida por la configuración inicial donde se encuentran los parámetros más importantes a tener en cuenta para empezar la rehabilitación con cada paciente, como es el caso del "Tamaño de los pies", dígase tanto ancho como largo de los mismos, así como la configuración de la "Orientación de los pies" y "Características de los pasos". Además, cuenta con el manejo de los botones "Inicio", "Pausa" y "Detener" que son los encargados de la administración de la marcha en esta interfaz, así como otras opciones de igual importancia, por ejemplo: la "Ayuda", la opción "Ver", donde se visualizan los escenarios, la configuración de los obstáculos y la opción "Archivo", que permite guardar los datos de los pacientes entre algunas otras configuraciones más.

3.2 Validación de la Solución

Una vez desarrollada la solución, es indispensable someterla a un proceso de pruebas con el objetivo de eliminar la mayor cantidad de errores posibles que pueda presentar y así obtener finalmente una aplicación con la calidad requerida. Las pruebas son tan importantes como el desarrollo.

El proceso de pruebas de esta investigación está regido por la metodología de desarrollo empleada: XP. Las pruebas en XP se dividen en dos grupos: las pruebas unitarias y las de aceptación.

3.2.1 Tipos de pruebas

Pruebas Unitarias:

Son pruebas que se escriben antes que el código. Las historias de usuario se dividen en tareas, las cuales representan las unidades principales de implementación, para cada una de ellas se diseña una prueba que debe pasar satisfactoriamente antes de la siguiente entrega del sistema. Cada tarea genera una o más pruebas de unidad que validan la implementación descrita en ella. Las pruebas se escriben como un componente ejecutable antes de que se implemente la tarea, el cual debe ser una aplicación independiente, debe simular el envío de la entrada a probar y verificar que el resultado cumple la especificación de salida (42).

Pruebas de Aceptación:

Este tipo de pruebas son realizadas por el cliente, como su nombre lo indica son pruebas para verificar la aceptación del usuario. Se diseñan de modo tal que se pruebe cada una de las historias de usuario definidas, para lo cual se deben especificar diferentes escenarios con diferentes condiciones y el cliente comprueba si obtuvo el resultado deseado. En caso de fallos, es necesario registrarlos en una planilla de no conformidades para resolverlas en el menor tiempo posible (43). Las pruebas de aceptación son consideradas como “pruebas de caja negra”.

Pruebas de Regresión:

Las pruebas de regresión no pertenecen a las pruebas definidas por la metodología XP, sin embargo, se pueden usar en conjunto a estas con el fin de detectar errores en el sistema que se hayan originado en la resolución de no conformidades luego de haber aplicado al menos una vez las pruebas. Generalmente modificaciones en un software generan nuevos errores, por tanto, estas pruebas se aplican después de la corrección de un error o después de la adición de nuevas funcionalidades con el objetivo de verificar que ningún defecto se añadió al sistema a raíz de la modificación (44).

De los tipos de técnicas descritas se seleccionaron para la validación del sistema correspondiente a este trabajo de diploma las pruebas de aceptación y de regresión para verificar que se satisfacen los requisitos establecidos por el usuario correctamente, teniendo en cuenta que el desarrollo de la solución está enfocado en una programación ágil y la entrega temprana de su primera versión funcional.

3.2.2 Técnicas de pruebas

Según Pressman *“Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación”*. El software debe probarse desde dos perspectivas diferentes: la lógica interna del programa (caja blanca)

y los requisitos del software, se comprueban utilizando técnicas de diseño de casos de prueba (caja negra) (16).

Pruebas de Caja Blanca:

La prueba de caja blanca se basa en el diseño de casos de prueba que usan la estructura de control del diseño procedimental para derivarlos, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. En el componente creado, se aplicó la técnica de pruebas de caja blanca del camino básico, puesto que permite obtener una medida de la complejidad lógica de un diseño y usarla como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Los casos de prueba derivados del camino básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa (16).

Pruebas de Caja Negra:

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se rige por su especificación, por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro, o sea, según (Pressman) se centran en los requisitos funcionales. Son pruebas sobre la interfaz del software. Enfocadas en las entradas y salidas y no en el código fuente (16).

3.2.3 Niveles de pruebas

Según Pressman el proceso de ingeniería del software se puede ver como una espiral. Inicialmente, la ingeniería de sistemas define el papel del software y conduce al análisis de los requisitos del software, donde se establece el dominio de información: la función, el comportamiento, el rendimiento, las restricciones y los criterios de validación del mismo. Una vez realizada la implementación se decide comenzar a realizar las pruebas oportunas para el control de la calidad del producto final, para ello se debe tener en cuenta los distintos niveles de pruebas. Los niveles de pruebas no son más que diferentes ángulos de verificar y validar un producto de software (16).

Niveles de Pruebas:

- Pruebas Unitarias.
- Pruebas de Componentes/ Pruebas de Integración.

- Pruebas de Funcionalidad.
- Pruebas de Sistema.
- Pruebas de Aceptación.
- Pruebas de Instalación.

Durante la realización de las pruebas al sistema se tuvo en cuenta los niveles según el orden en que se fueron realizando cada una de estas, que coincide con el mismo orden en que se han mencionado anteriormente. Aunque en ocasiones no se documente los detalles de cada prueba realizada, una correcta validación siempre debe cumplir con dichos niveles para asegurar que el software obtenido cumple con la calidad requerida.

3.3 Pruebas realizadas al sistema

Se realizan pruebas de Caja Negra y Pruebas de Caja Blanca con el fin de probar las funcionalidades de la solución propuesta.

3.3.1 Pruebas de Caja Blanca

En el caso del algoritmo diseñado, cuyo código se muestra en los anexos (Ver anexo 1), la aplicación de la prueba del camino básico quedó de la siguiente manera:

1. Caminos posibles identificados a partir del código fuente:

   Camino 1: (1-4-7-11-14)

   Camino 2: (1-4-8-11-14)

   Camino 3: (1-2-5-12-14)

   Camino 4: (1-2-5-9-12-14)

   Camino 5: (1-2-3-6-13-14)

   Camino 6: (1-2-3-6-10-13-14)

   Camino 7: (1-2-3-14)

2. Grafo Asociado

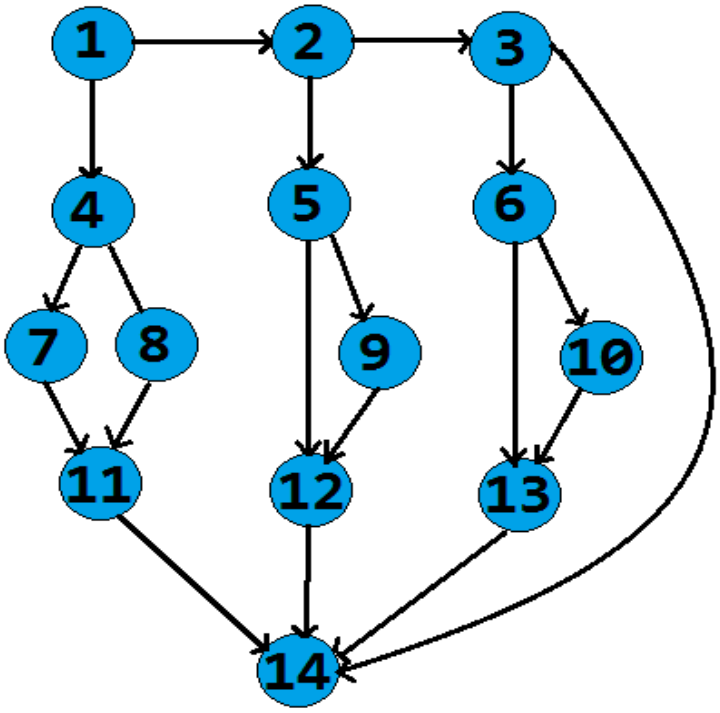

Fig. 14. Grafo Asociado al Código Fuente del Anexo.

3. Se calcula la complejidad ciclomática del grafo:

La complejidad ciclomática V (G), se define como:

V (G) = A - N + 2 donde A es el número de aristas del grafo y N el número de nodos, por tanto:

V(G) = 19-14 + 2

V(G) = 7

La complejidad ciclomática no solo puede ser calculada mediante la fórmula antes expuesta, puesto que también se define como:

V(G) = P + 1 donde P es el número de nodos predicados contenidos en el grafo G .Los nodos predicados son aquellos que se caracterizan porque dos o más aristas emergen de él , por tanto :

V(G) = 6 +1

V(G) = 7

Los cálculos obtenidos con anterioridad dejan en evidencia que el grafo contiene 7 regiones, por tanto, la complejidad ciclomática del grafo de flujo de la figura es siete, lo que conlleva a que este sea el

número de caminos independientes del conjunto básico del código y el límite superior de pruebas que se deben realizar para asegurarse que se ejecute cada sentencia al menos una vez. Como son dos caminos para dos soluciones, realizamos dos pruebas de camino básico, una para el camino que lleva a la solución exitosa y otro para donde la solución es fallida.

4. Se preparan los casos de prueba que obliguen la ejecución de cada camino básico:

Tabla 6. Caso de Prueba para el Camino Básico 1

Caso de prueba para el camino básico 1	
Descripción: El sistema debe mostrar un paso detrás de otro en el entorno virtual, o iniciar la marcha.	
Condición de ejecución: El paciente debe estar en el estado “Running”, o sea corriendo o caminando.	
Procesamiento de la prueba	
Dato de entrada:	State
Tipo de dato esperado:	State
Resultados esperados:	

Tabla 7. Caso de Prueba para el Camino Básico 2.

Caso de prueba para el camino básico 2
Descripción: El sistema debe mostrar un paso detrás de otro en el entorno virtual, o iniciar la marcha.
Condición de ejecución: El paciente debe estar en el estado “Running”, o sea corriendo o caminando.

Procesamiento de la prueba	
Dato de entrada:	State
Tipo de dato esperado:	State
Resultados esperados:	El Sistema mantiene el entorno virtual apagado debido a que no se está realizando ninguna acción sobre él.

3.3.2 Pruebas de Caja Negra

Dentro de la técnica de caja negra el método que se seleccionó para realizar las pruebas en este caso es el método de la partición de equivalencia. Este método consiste en dividir el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente a una prueba realizada con cualquier otro valor de dicha clase, lo cual quiere decir que si el caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error y viceversa (16). A continuación, se muestra en la Tabla 8 el caso de prueba diseñado para la historia de usuario “Configurar Obstáculos”.

Tabla 8. Caso de Prueba "Configurar Obstáculos"

Escenario	Descripción	Variables	Respuesta del sistema	Flujo central
EC 1.1 Configurar obstáculos con datos correctos.	Los obstáculos son adicionados correctamente.	Largo: 20cm Tiempo: 20 segundos.	El sistema muestra el mensaje “Obstáculos adicionados correctamente”.	1-Desplegar el menú “Ver”. 2-Seleccionar la opción de menú “Configurar parámetros de los obstáculos”. 3-Introducir los datos. 4-Seleccionar la opción “Adicionar”.
EC 1.2 Configurar obstáculos con datos incorrectos.	Los obstáculos no son adicionados.	Largo: 200cm Tiempo: 20 segundos.	El sistema muestra el mensaje “Los obstáculos no pudieron ser adicionados”.	1-Desplegar el menú “Ver”. 2-Seleccionar la opción de menú “Configurar parámetros de los obstáculos”. 3-Introducir los datos. 4-Seleccionar la opción “Adicionar”.
EC 1.3 Configurar obstáculos con datos nulos.	Los obstáculos no son adicionados.	Largo: nulo Tiempo: nulo	El sistema muestra el mensaje “Existen campos vacíos”.	1-Desplegar el menú “Ver”. 2-Seleccionar la opción de menú “Configurar parámetros de los obstáculos”. 3-Introducir los datos. 4-Seleccionar la opción “Adicionar”.

En la Tabla 9 se muestran las variables para el caso de prueba “Configurar Obstáculos” necesarias para poder probar cada uno de los escenarios de dicho caso de prueba.

Tabla 9. Variables para el caso de prueba "Configurar Obstáculos".

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
Variable 1	Largo	Campo numérico	No	Dimensión que se va a adicionar al obstáculo.
Variable 2	Tiempo	Campo numérico	No	Cantidad de intervalo de tiempo entre cada obstáculo.

Luego de la ejecución de las pruebas al software, se registran los errores detectados para corregirlos posteriormente. Todos estos errores son almacenados en una planilla de no conformidades. En la Tabla 10 se muestran las no conformidades recogidas en el proceso de prueba realizado para la historia de usuario “Configurar Obstáculos”.

Tabla 10. No Conformidades del Caso de Prueba "Configurar Obstáculo"

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasif.	Estado NC	Respuesta del Equipo de Desarrollo
Aplicación	1	El campo “Largo” acepta medidas muy largas.	Adicionar obstáculos	Prueba	No Significativa	20/03/2017 PD 21/03/2017 RA	Se corrigió el error encontrado, el sistema ya no admite medidas muy largas.
Aplicación	2	El campo “Tiempo” acepta intervalos muy cortos.	Adicionar obstáculos	Prueba	No Significativa	28/03/2017 PD 29/03/2017 RA	Se corrigió el error encontrado, el sistema no admite intervalos de tiempo muy cortos.
Aplicación	3	El sistema no adiciona nuevos valores de los obstáculos una vez insertados.	Adicionar obstáculos	Prueba	Significativa	2/04/2017 PD 2/04/2017 RA	Se corrigió el error encontrado, el sistema adicionó los nuevos valores insertados.

3.4 Análisis de los resultados de las pruebas aplicadas

Luego de la ejecución de las pruebas realizadas a cada historia de usuario, en un total de cuatro iteraciones, las mismas arrojaron las siguientes no conformidades que se muestran en la Fig. 15. En la primera iteración existieron un total de catorce no conformidades divididas de la siguiente forma: cinco de ortografía, tres de errores de interfaz, una de funcionalidad y cinco de validación. Luego de un proceso de depuración se realizó una segunda iteración, arrojando las siguientes no conformidades: dos de ortografía, una de interfaz y cuatro de validación. Se realizó una tercera iteración, en la que se detectaron: una de ortografía y una de validación y se finalizó en una cuarta iteración donde el sistema quedó libre de errores, obteniéndose un software que cumple con la calidad requerida.

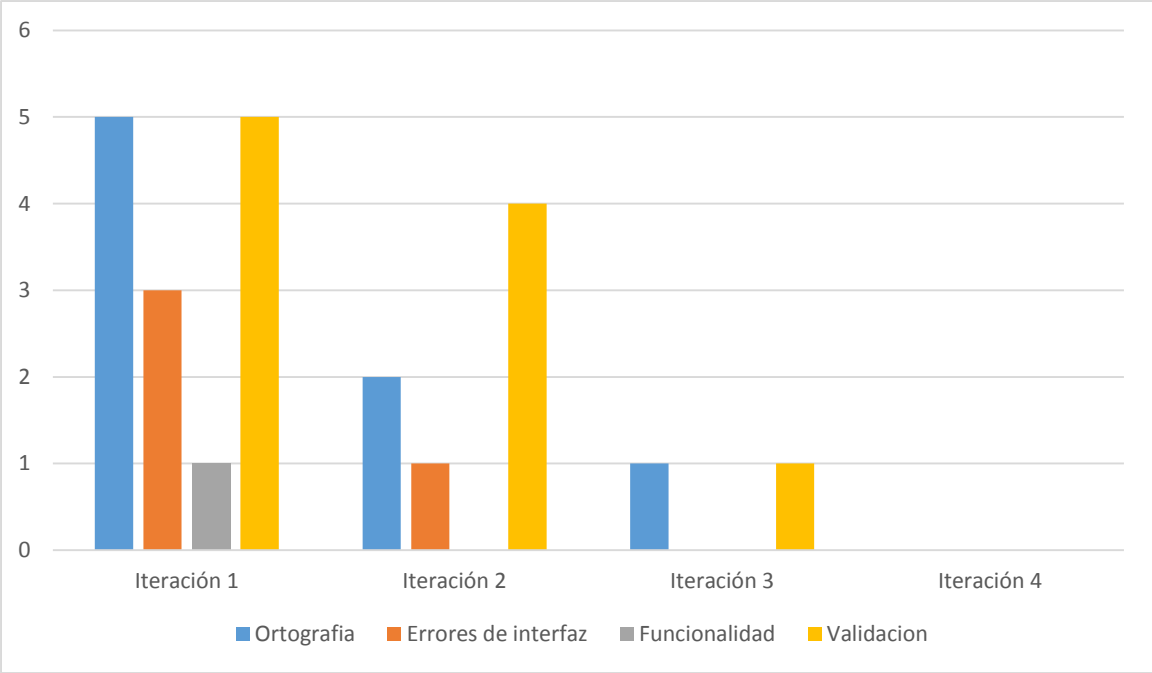

Fig. 15. Resultados de las pruebas aplicadas.

3.5 Conclusiones del Capítulo

Durante el desarrollo de este capítulo se realizó la implementación del código, usando como base las tareas de ingeniería previamente definidas, 34 exactamente. En dicha implementación se emplearon diferentes estándares de codificación para la adecuada escritura del código, facilitándose la comprensión del mismo.

Se diseñaron 15 casos de prueba para validar las 15 historias de usuario definidas en el capítulo 2. Se llevó a cabo el proceso de validación empleando las pruebas de aceptación y las de regresión, mediante las técnicas de caja blanca (método del camino básico) y caja negra (método de partición de equivalencia). Se registraron los errores detectados, los cuales finalmente fueron corregidos, obteniéndose una versión del sistema lista para trabajar correctamente.

Conclusiones

Una vez finalizado el desarrollo de esta investigación, cumpliéndose los objetivos trazados al inicio, se puede concluir lo siguiente:

1. El análisis realizado sobre las herramientas de rehabilitación virtual que existen actualmente, aportó los elementos necesarios para el desarrollo del nuevo Sistema de Configuraciones Dinámicas y Ambientes Virtuales para el Rehabilitador de Marcha.
2. Se realizó el diseño de la solución utilizando los artefactos definidos por XP, metodología seleccionada para el desarrollo del sistema y se obtuvo como resultado 15 historias de usuario, de las que se especificaron 15 requisitos funcionales y 17 no funcionales, que fueron cumplidos en su totalidad.
3. Se implementó un componente para las configuraciones dinámicas y ambientes virtuales del Rehabilitador de Marcha, dándole así solución al problema planteado inicialmente.
4. Se llevó a cabo el proceso de validación del sistema mediante las pruebas de aceptación y regresión, usando las técnicas de caja blanca y caja negra y los métodos de camino básico y partición de equivalencia, respectivamente. Se obtuvo como resultado un total de 23 no conformidades, las cuales fueron resueltas satisfactoriamente, finalizándose el proceso con un sistema listo para entregar al cliente.

Recomendaciones

Al término del presente trabajo de diploma se recomienda:

1. Incrementar mejoras en los entornos virtuales añadidos, así como agregar otros entornos virtuales que se consideren de acorde para la terapia.
2. Definir configuraciones personalizadas a los obstáculos teniendo en cuenta el nivel de rehabilitación al que se está sometiendo cada paciente, estas pudiesen estar divididas en fáciles, medias y difíciles.
3. Añadir obstáculos de otra índole para incrementar la variedad en los entornos virtuales y fomentar un mayor entretenimiento al paciente.

Referencias

1. **Salcedo, Miguel Angel Ortiz**. *InTrainer, Sistema de rehabilitación cardiaca aumentado por realidad virtual*. Universidad del País Vasco : Euskal Herriko Unibertsitatea, 2010.

2. **Moreno F., Ojeda J., Ramírez J., Mena C., Rodríguez O., Rangel J., Alvarez S**. *Un framework para la rehabilitación física en Miembros.* Universidad Central de Venezuela Caracas : Primera Conferencia Nacional de Computación, Informática y Sistemas, Centro de computación gráfica, escuela de computación, 2013.

3. **Miller, Gómez M.** *Aplicación de realidad virtual en la rehabilitación cognitiva.* número enero –junio 2013, s.l. : Revista Vínculos, 2013, Vol. Vol. 10.

4. **Pedraza, Daniel Fernández-Áviles**. *Evaluación y Tratamiento basados en Realidad Virtual de la Negligencia Espacial Unilateral: Análisis del Estado de la Cuestión y Demostración de Enfoques Alternativos*. Madrid : Universidad Politécnica de Madrid, Facultad de Informática, 2014.

5. **Alcides Alvear Suárez, Graciela E. Quintero Ramírez**. *Ambientes virtuales para rehabilitación física y cognitiva*. [En línea] 23-27 de Julio de 2012. http://www.laccei.org/LACCEI2012-Panama/RefereedPapers/no%20cw/RP060.pdf.

6. **Universidad Militar Nueva Granada**, Facultad Estudios a Distancia. *Ambientes virtuales*. [En línea] 16 de Julio de 2015. http://www.umng.edu.co/ambientes-virtuales.

7. **Pérez-Salas, Claudia P**. *Realidad Virtual: Un Aporte Real para la Evaluación y el Tratamiento de Personas con Discapacidad Intelectual*. [En línea] Diciembre de 2008. http://www.scielo.cl/scielo.php?pid=S0718-48082008000200011&script=sci_arttext.

8. **Martín, Abigail**. *Una terapia divertida: juegos de realidad virtual y fisioterapia. Fisioterapia Online.* [En línea] 2017. https://www.fisioterapia-online.com/articulos/una-terapia-divertida-juegos-de-realidad-virtual-y-fisioterapia.

9. Entornos virtuales en rehabilitacion. *Introducción Ing. Civil en Computación mención Informática.* [En línea] 9 de Junio de 2014. https://blogmane.wordpress.com/2014/04/22/entornos-virtuales-en-rehabilitacion/.

10. **Pombo, Alejandro**. *Realidad Virtual en la Rehabilitación ¿Ventaja o Desventaja? Rehabilitación Virtual.* [En línea] 2015. https://sites.google.com/a/correo.unimet.edu.ve/realidad-virtual-rehabilitacion/pagina.

11. **Paez, Andy Hernández**. *Especificación de Requisitos de Software del Proyecto Rehabilitador de Marcha*. La Habana : Universidad de las Ciencias Informáticas, 2014.

12. **Roig, Josep**. *Fabricantes de cintas de correr*. [En línea] 6 de Junio de 2015. Cintasdecorrer.com.

13. **Dora.** *Beam, el proyector androide escondido en una bombilla.* [En línea] 2017. http://es.engadget.com/2015/02/18/beam-proyector-androide-bombilla-video/.

14. Universidad Nacional Autónoma de México, Facultad de Contaduría y Administración. Apuntes Digitales Plan 2012. *Licenciatura en Informática.* [En línea] 2012. http://fcasua.contad.unam.mx/apuntes/interiores/docs/2012/informatica/1/1169.pdf.

15. **Acebes, María**. *Pantalla LED,televisor CRT, televisor de plasma y televisor TFT-LCD*. [En línea] 17 de Octubre de 2014. https://mariaacebes.wordpress.com/2014/10/17/pantalla-ledtelevisor-crt-televisor-de-plasma-y-televisor-tft-lcd/.

16. **Pressman, Roger S**. *Ingeniería de Software, Un Enfoque Práctico*, Séptima Edición. [En línea] 2002. http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF.

17. **Yolanda, López B**. *Metodo Ágil de Desarrollo de Software-XP*. [En línea] 2014. http://www.runayupay.org/publicaciones/2244_555_COD_18_290814203015.pdf.

18. **Isama, Danles**. *UML Método o Lenguaje de Modelado*. [En línea] 25 de Octubre de 2012. https://prezi.com/vxo07phkponh/uml/.

19. *¿Qué es y para qué sirve UML? Versiones de UML (Lenguaje Unificado de Modelado). Tipos de diagramas UML*. *aprenderaprogramar.com.* [En línea] http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46:lenguajes-y-entornos&Itemid=163.

20. *Descripción de ArgoUML*. [En línea] 2014. http://www.softpedia.es/programa-ArgoUML.

21. *Exposición de Poseidon*. [En línea] 2014. http://es.scribd.com/doc/68588068/EXPOSICION-DE-POSEIDON.

22. *Lenguajes de Programación*. *Future Lives.* [En línea] 2017. https://futurelives.jimdo.com/travel-technology/lenguajes-de-programacion-clasificacion-ejemplos/.

23. **Flores, Linda I. Olivares**. *Manual de Programaci´on en Lenguaje C+.* [En línea] 2008. http://paginas.matem.unam.mx/pderbf/images/mprogintc++.pdf.

24. **Pellicer, Linares J**. El Lenguaje C#. *Programación amb C# .NET.* [En línea] http://users.dsic.upv.es/~jlinares/csharp/Tema%203.pdf.

25. **Torres, Jesús**. *Proyecto Qt: Framework de desarrollo de aplicaciones*. [En línea] 28 de Enero de 2013. https://medium.com/jmtorres/proyecto-qt-framework-de-desarrollo-de-aplicaciones-2b2f895ac285.

26. Microsoft. Introducción a .NET Framework. *Microsoft Developer Network.* [En línea] 2017. https://msdn.microsoft.com/es-es/library/hh425099(v=vs.110).aspx.

27. **Cambiaso, Diego**. *Qt Creator – Completo entorno de desarrollo multiplataforma*. [En línea] 10 de Junio de 2010. http://pixelcoblog.com/qt-creator-completo-entorno-de-desarrollo-multiplataforma/.

28. **Iglesias, César**. *Indroducción a Monodevelop*. *Lignux.* [En línea] 14 de Septiembre de 2013. https://lignux.com/indroduccion-a-monodevelop/.

29. **Candil, Dani**. *Unity, el motor de desarrollo capaz de partir la historia de los videojuegos en dos*. [En línea] Octubre de 2011. https://www.vidaextra.com/industria/unity-el-motor-de-desarrollo-capaz-de-partir-la-historia-de-los-videojuegos-en-dos.

30. **Larman, C**. *UML y Patrones.* 2003.

31. *Historia de usuario, Creación de historias de usuario, Ejemplos, Uso, Beneficios, Mapas Historia, Limitaciones, Las historias de usuario y casos de uso.* 2013.

32. *Artefactos XP.* 2014.

33. **Cevallos, Karla**. *Ingeniería del Software*. [En línea] Julio de 2015. https://ingsotfwarekarlacevallos.wordpress.com/author/karlacevallos/.

34. *Introducción a la Arquitectura del Software.* 2013.

35. **Reynoso, C; Kiccillof, N**. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. [En línea] Marzo de 2004. http://cic.javerianacali.edu.co/wiki/lib/exe/fetch.php?media=materias:estiloypatron.pdf.

36. **Valdezate, Alejandro Sánchez**. *Modelado y Gestión de la Variabilidad en Sistemas Software*. [En línea] 2006. http://alejandro.valdezate.net/wp-content/uploads/2013/08/memoria_urjc_lps.pdf.

37. **Tedeschi, N**. *¿Qué es un patrón de diseño?* [En línea] 2014. http://msdn.microsoft.com/eses/library/bb972240.aspx.

38. **Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.** *Design Patterns: Elements of reusable object-oriented software.* s.l. : Addison-Wesley, 1995.

39. **Gascon, Ulises**. *Curso-JS-Avanzado-para-desarrolladores-Front-end_ed2*. [En línea] 29 de Marzo de 2017. https://github.com/Fictizia/Curso-JS-Avanzado-para-desarrolladores-Front-end_ed2/blob/master/teoria/clase15.md.

40. Artefactos. *in Programación Extrema.* 2014.

41. *Guía para implementar estándares de codificación.*

42. **Crispin, L**. *Testing Extreme Programming.* 2003.

43. **J.J.E. Gutiérrez, J. M, M. M, J. T**. *Pruebas del sistema en Programación Extrema.*

44. **Fernández-Pello, J.** *Pruebas de Regresión.* s.l. : in SoftQaNetwork, 2014.

45. **Moxo, Beatriz Adriana Sabino** *Rehabilitación virtual mediante interfaces naturales de usuario.*. 7467, México DF : Revista Iberoamericana para la Investigación y el Desarrollo Educativo, 2014, Vols. ISSN 2007 - 7467. 7467.

46. **Carro, Juan Pablo**. *Resumen de Patrones*. *Scribd.* [En línea] https://es.scribd.com/document/98414166/Resumen-de-Patrones.

Bibliografía

1. **Salcedo, Miguel Angel Ortiz**. *InTrainer, Sistema de rehabilitación cardiaca aumentado por realidad virtual*. Universidad del País Vasco : Euskal Herriko Unibertsitatea, 2010.

2. **Moreno F., Ojeda J., Ramírez J., Mena C., Rodríguez O., Rangel J., Alvarez S**. *Un framework para la rehabilitación física en Miembros.* Universidad Central de Venezuela Caracas : Primera Conferencia Nacional de Computación, Informática y Sistemas, Centro de computación gráfica, escuela de computación, 2013.

3. **Miller, Gómez M.** *Aplicación de realidad virtual en la rehabilitación cognitiva.* número enero –junio 2013, s.l. : Revista Vínculos, 2013, Vol. Vol. 10.

4. **Pedraza, Daniel Fernández-Áviles**. *Evaluación y Tratamiento basados en Realidad Virtual de la Negligencia Espacial Unilateral: Análisis del Estado de la Cuestión y Demostración de Enfoques Alternativos*. Madrid : Universidad Politécnica de Madrid, Facultad de Informática, 2014.

5. **Alcides Alvear Suárez, Graciela E. Quintero Ramírez**. *Ambientes virtuales para rehabilitación física y cognitiva*. [En línea] 23-27 de Julio de 2012. http://www.laccei.org/LACCEI2012-Panama/RefereedPapers/no%20cw/RP060.pdf.

6. **Universidad Militar Nueva Granada**, Facultad Estudios a Distancia. *Ambientes virtuales*. [En línea] 16 de Julio de 2015. http://www.umng.edu.co/ambientes-virtuales.

7. **Pérez-Salas, Claudia P**. *Realidad Virtual: Un Aporte Real para la Evaluación y el Tratamiento de Personas con Discapacidad Intelectual*. [En línea] Diciembre de 2008. http://www.scielo.cl/scielo.php?pid=S0718-48082008000200011&script=sci_arttext.

8. **Martín, Abigail**. *Una terapia divertida: juegos de realidad virtual y fisioterapia. Fisioterapia Online.* [En línea] 2017. https://www.fisioterapia-online.com/articulos/una-terapia-divertida-juegos-de-realidad-virtual-y-fisioterapia.

9. Entornos virtuales en rehabilitacion. *Introducción Ing. Civil en Computación mención Informática.* [En línea] 9 de Junio de 2014. https://blogmane.wordpress.com/2014/04/22/entornos-virtuales-en-rehabilitacion/.

10. **Pombo, Alejandro**. *Realidad Virtual en la Rehabilitación ¿Ventaja o Desventaja? Rehabilitación Virtual.* [En línea] 2015. https://sites.google.com/a/correo.unimet.edu.ve/realidad-virtual-rehabilitacion/pagina.

11. **Paez, Andy Hernández**. *Especificación de Requisitos de Software del Proyecto Rehabilitador de Marcha*. La Habana : Universidad de las Ciencias Informáticas, 2014.

12. **Roig, Josep**. *Fabricantes de cintas de correr*. [En línea] 6 de Junio de 2015. Cintasdecorrer.com.

13. **Dora.** *Beam, el proyector androide escondido en una bombilla.* [En línea] 2017. http://es.engadget.com/2015/02/18/beam-proyector-androide-bombilla-video/.

14. Universidad Nacional Autónoma de México, Facultad de Contaduría y Administración. Apuntes Digitales Plan 2012. *Licenciatura en Informática.* [En línea] 2012. http://fcasua.contad.unam.mx/apuntes/interiores/docs/2012/informatica/1/1169.pdf.

15. **Acebes, María**. *Pantalla LED,televisor CRT, televisor de plasma y televisor TFT-LCD*. [En línea] 17 de Octubre de 2014. https://mariaacebes.wordpress.com/2014/10/17/pantalla-ledtelevisor-crt-televisor-de-plasma-y-televisor-tft-lcd/.

16. **Pressman, Roger S**. *Ingeniería de Software, Un Enfoque Práctico*, Séptima Edición. [En línea] 2002. http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF.

17. **Yolanda, López B**. *Metodo Ágil de Desarrollo de Software-XP*. [En línea] 2014. http://www.runayupay.org/publicaciones/2244_555_COD_18_290814203015.pdf.

18. **Isama, Danles**. *UML Método o Lenguaje de Modelado*. [En línea] 25 de Octubre de 2012. https://prezi.com/vxo07phkponh/uml/.

19. *¿Qué es y para qué sirve UML? Versiones de UML (Lenguaje Unificado de Modelado). Tipos de diagramas UML*. *aprenderaprogramar.com.* [En línea] http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46:lenguajes-y-entornos&Itemid=163.

20. *Descripción de ArgoUML*. [En línea] 2014. http://www.softpedia.es/programa-ArgoUML.

21. *Exposición de Poseidon*. [En línea] 2014. http://es.scribd.com/doc/68588068/EXPOSICION-DE-POSEIDON.

22. *Lenguajes de Programación*. *Future Lives.* [En línea] 2017. https://futurelives.jimdo.com/travel-technology/lenguajes-de-programacion-clasificacion-ejemplos/.

23. **Flores, Linda I. Olivares**. *Manual de Programaci´on en Lenguaje C+.* [En línea] 2008. http://paginas.matem.unam.mx/pderbf/images/mprogintc++.pdf.

24. **Pellicer, Linares J**. El Lenguaje C#. *Programación amb C# .NET.* [En línea] http://users.dsic.upv.es/~jlinares/csharp/Tema%203.pdf.

25. **Torres, Jesús**. *Proyecto Qt: Framework de desarrollo de aplicaciones*. [En línea] 28 de Enero de 2013. https://medium.com/jmtorres/proyecto-qt-framework-de-desarrollo-de-aplicaciones-2b2f895ac285.

26. Microsoft. Introducción a .NET Framework. *Microsoft Developer Network.* [En línea] 2017. https://msdn.microsoft.com/es-es/library/hh425099(v=vs.110).aspx.

27. **Cambiaso, Diego**. *Qt Creator – Completo entorno de desarrollo multiplataforma*. [En línea] 10 de Junio de 2010. http://pixelcoblog.com/qt-creator-completo-entorno-de-desarrollo-multiplataforma/.

28. **Iglesias, César**. *Indroducción a Monodevelop*. *Lignux.* [En línea] 14 de Septiembre de 2013. https://lignux.com/indroduccion-a-monodevelop/.

29. **Candil, Dani**. *Unity, el motor de desarrollo capaz de partir la historia de los videojuegos en dos*. [En línea] Octubre de 2011. https://www.vidaextra.com/industria/unity-el-motor-de-desarrollo-capaz-de-partir-la-historia-de-los-videojuegos-en-dos.

30. **Larman, C**. *UML y Patrones.* 2003.

31. *Historia de usuario, Creación de historias de usuario, Ejemplos, Uso, Beneficios, Mapas Historia, Limitaciones, Las historias de usuario y casos de uso.* 2013.

32. *Artefactos XP.* 2014.

33. **Cevallos, Karla**. *Ingeniería del Software*. [En línea] Julio de 2015. https://ingsotfwarekarlacevallos.wordpress.com/author/karlacevallos/.

34. *Introducción a la Arquitectura del Software.* 2013.

35. **Reynoso, C; Kiccillof, N**. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. [En línea] Marzo de 2004. http://cic.javerianacali.edu.co/wiki/lib/exe/fetch.php?media=materias:estiloypatron.pdf.

36. **Valdezate, Alejandro Sánchez**. *Modelado y Gestión de la Variabilidad en Sistemas Software*. [En línea] 2006. http://alejandro.valdezate.net/wp-content/uploads/2013/08/memoria_urjc_lps.pdf.

37. **Tedeschi, N**. *¿Qué es un patrón de diseño?* [En línea] 2014. http://msdn.microsoft.com/eses/library/bb972240.aspx.

38. **Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.** *Design Patterns: Elements of reusable object-oriented software.* s.l. : Addison-Wesley, 1995.

39. **Gascon, Ulises**. *Curso-JS-Avanzado-para-desarrolladores-Front-end_ed2*. [En línea] 29 de Marzo de 2017. https://github.com/Fictizia/Curso-JS-Avanzado-para-desarrolladores-Front-end_ed2/blob/master/teoria/clase15.md.

40. Artefactos. *in Programación Extrema.* 2014.

41. *Guía para implementar estándares de codificación.*

42. **Crispin, L**. *Testing Extreme Programming.* 2003.

43. **J.J.E. Gutiérrez, J. M, M. M, J. T**. *Pruebas del sistema en Programación Extrema.*

44. **Fernández-Pello, J.** *Pruebas de Regresión.* s.l. : in SoftQaNetwork, 2014.

45. **Moxo, Beatriz Adriana Sabino** *Rehabilitación virtual mediante interfaces naturales de usuario.*. 7467, México DF : Revista Iberoamericana para la Investigación y el Desarrollo Educativo, 2014, Vols. ISSN 2007 - 7467. 7467.

46. **Carro, Juan Pablo**. *Resumen de Patrones*. *Scribd.* [En línea] https://es.scribd.com/document/98414166/Resumen-de-Patrones.

Glosario de Términos

Framework: es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Ergométrica: de Ergometría, que significa Medida del trabajo o del esfuerzo llevado a cabo por los músculos.

Plugins: un complemento o plug-in es una aplicación (o programa informático) que se relaciona con otra para agregarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la interfaz de programación de aplicaciones.

Interoperabilidad: la habilidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

Breakpoints: punto de parada, punto de interrupción, punto de quiebre o parada dinámica. En programación, especialmente en depuración del programa, un punto de parada o breakpoint, es una pausa intencional y controlada durante la ejecución de un programa.

Submódulo: un módulo que forma parte de un módulo más grande. En programación, un programa puede estar compuesto por varios subprogramas o submódulos.

Anexos

Anexo 1: Fragmento del código fuente de la aplicación

```
public void SetState(State state)
        {
      stateM = state;
                switch (state)
                {
                case State.Started:
            if (!isRunning)
            {
                GoToNextPoint();
                ObsGoToNextPoint();
                isRunning = true;
                FeetWay.RunFeets();
            }
            else
            {
                __obsagent.Resume();
                __obsDeployer.StartPutting();
                __agent.Resume();
                FeetWay.RestoreFeets();
            }
            break;
                case State.Paused:
            if (isRunning)
            {
                __obsagent.Stop();
                __obsDeployer.StopPutting();
                __agent.Stop();
                FeetWay.StopFeets();
            }
```

```
    break;
        case State.Stopped:
    if (isRunning)
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
                break;
        }
}
```