Universidad de las Ciencias Informáticas

"Facultad 2"

Título: Desarrollo de un programa de interfaz entre el código de Monte Carlo MCNPX y las imágenes médicas corregistradas (SPECT-CT) en formato DICOM, para el cálculo de distribución de dosis en tejidos tumorales y sanos.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Yoandy Rey Díaz Nápoles.

Tutores: Dra. Maritza Rodríguez Gual (InSTEC)

Dr. Joaquín González González (INOR)

Ing. Yahima Vigo Valdes (UCI)

Datos de Contacto

Nombre de la tutora: Maritza Rodríguez Gual

Tels.: 8789857(trab), 6980171(casa)

E-mail: mrgual@instec.cu

La Dra. Maritza Rodríguez Gual, Investigadora Auxiliar del Departamento de Desarrollo Tecnológico (DDT), del Instituto Superior de Tecnologías y Ciencias Aplicadas (InSTEC) se graduó en la Facultad de Ciencias y Tecnología Nuclear (FCTN) con el Título de Ingeniero en Energética Nuclear en el año 1987 y que posteriormente, en el año 1990 obtiene el Título de Ingeniero Especialista en Técnica de Reactores Nucleares, en la Universidad Técnica de Budapest (UTB), Hungría. Tiene una experiencia en la actividad científica de 24 años.

Ha desempeñado su actividad de investigación en los temas de: la Física de Reactores Nucleares, la Física Médica, Dosimetría, Protección Radiológica, Seguridad Nuclear, Blindaje de la radiación, Biofísica y Nanociencia, con una producción científica de 44 publicaciones, entre las cuales se destacan revistas de impacto internacional como: la TRANSACTIONS of American Nuclear Society, la Annals of Nuclear Energy, la Journal of Applied Radiation and Isotopes, la Medical Physics, la Brazilian Journal of Physics, la Revista Española de Medicina Nuclear, la World Journal of Nuclear of Medicine, la Journal of Biological Physics y la Fresenius Environmental Bulletin. También ha publicado en revistas nacionales de visibilidad internacional como: la Revista Cubana de Física, la Revista de Ingeniería Mecánica de la CUJAE y la Revista de Ciencias Biológicas del CENIC. Ha participado en 36 eventos nacionales e internacionales, en Proyectos Internacionales del Organismo Internacional de Energía Atómica (IAEA) y de la Fundación de Amparo a la Pesquisa del estado de Sao Paulo (FAPESP), en proyectos de Programas Ramales Nucleares (PRN), Proyectos No Asociados a Programas (PNAP) y además ha sido Jefa de proyectos de investigación PNAP y PRN, obteniendo resultados destacados a nivel de Instituto y de Agencia. Participó en 2 tareas técnicas nacionales. Pertenece a la Sociedad Cubana de Oncología, Radioterapia y Medicina Nuclear y a la Sociedad Cubana de Física.

Ha recibido estudios de post-grado en el Centro Internacional de Física Teórica (ICTP), Trieste, Italia, en el Instituto de Pesquisas Energéticas y Nucleares (IPEN), Sao Pablo, Brasil, mediante una Beca MANPOWER del IAEA y también participó en el programa de doctorado cooperativo entre el ICTP-CLAF(Consejo Latinoamericano de Física), en Brasil. Ha participado en congresos internacionales en países como

Portugal y Brasil. Ha sido miembro del Consejo Técnico de la Unidad Científico Técnica del InSTEC, miembro del Tribunal de la Maestría de Física Médica y es miembro de la Sociedad Cubana de Física Médica.

Ha impartido clases en la Asignatura de Instalaciones Nucleares y Radiactivas, Complementos de Física Nuclear y Técnicas Nucleares en el InSTEC y de Física II en la Universidad de Ciencias Informáticas (UCI). Ha tutorado varios trabajos presentados en Forum Estudiantiles y se encuentra tutorando 2 tesis de grado. Ha impartido cursos de post-grado en el Instituto Nacional de Oncología y Radiobiología (INOR), Hospital Hermanos Ameijeiras (HHA) y fue profesora de la Maestría de Física Médica. Impartió tres seminarios internacionales: uno en el Instituto de Pesquisas Energéticas e Nucleares (IPEN), de Sao Paulo y uno en el Instituto de Física (IF), Universidad de São Paulo (USP) Brasil. Además, ha sido profesora de cursos internacionales.

Nombre del tutor: Joaquín Jorge González González.

El Dr. Joaquín Jorge González González, Profesor Auxiliar Facultad de Medicina "Manuel Fajardo", Universidad de La Habana. Se graduó en Universidad Estatal de Odesa, Ucraina con el Título de Físico en el año 1983 y que posteriormente, obtiene el Título de Máster en Ciencias Físico-Matemáticas. Tiene una experiencia de 27 años en el Campo de la Medicina Nuclear tanto en la investigación como en la docencia. Trabaja en el Instituto Nacional de Oncología y Radiobiología (INOR) de Cuba. Es miembro del Grupo Nacional de Oncología. Ministerio de Salud Pública, Cuba y del Grupo Especial de Trabajo (GET) de Medicina Nuclear de la Unidad Nacional de Control de Cáncer (UNCC).Tiene 17 publicaciones (nacionales e internacionales) en los últimos 10 años entre las cuales se destacan revistas de impacto internacional como: la Journal of Nuclear Medicine, Revista Española de Medicina Nuclear y la World Journal of Nuclear of Medicine. También ha publicado en revistas nacionales de visibilidad internacional como: la Revista Cubana de Física. Es co-editor de dos libros publicados en España: Oncología Nuclear y Neuroimagen Nuclear. Ha desempeñado misiones técnicas de asesorías en la construcción y/o remodelación de las unidades de Medicina Nuclear del país y misiones de experto del IAEA para modernizar Cámaras Gamma en Colombia, Brasil y Paraguay. Pertenece a la Sociedad Cubana de Oncología, Radioterapia y Medicina Nuclear y a la Sociedad Cubana de Física. Ha desempeñado su actividad de investigación en aspectos dosimétricos y radiobiológicos de la terapia del cáncer con radionucleidos. Ha participado en proyectos de Programas Ramales Nucleares (PRN) y Proyectos No Asociados a Programas (PNAP). Ha impartido 7 cursos de post-grado a nivel nacional en los últimos 10 años. Ha recibido varios premios, distinciones y reconocimientos por resultado científico destacado a nivel de Academia de Ciencias de Cuba, en 1993, en Toronto, Canadá en el 2001y en Stgo de Chile. Chile en el 2002 y por la Sociedad Cubana de Farmacología en el 2007.

Nombre de la tutora: Ing. Yahima Vigo Valdes

Tels.: 8372143

E-mail: yvigo@uci.cu

“Si buscas resultados distintos, no hagas siempre lo mismo.”

Albert Einstein

Agradecimientos

Le agradezco mucho a mi novia, que siempre me ha apoyado, y ha sabido levantar mi ánimo en los momentos más difíciles. A todas mis amistades, tanto las que vienen del mismo pre-universitario como las conocidas aquí en la universidad, que no las menciono por ser demasiadas; gracias por ayudarme de una forma u otra. A mis tutores que me fueron de gran ayuda, a los profesores de los cuales recibí clases, que me enseñaron que no todos pensamos iguales, fortalecieron mi carácter, mi pensar y mi forma de actuar.

Dedicatoria

Dedico este estudio a toda mi familia, especialmente a mis padres que siempre se han sacrificado por mí, y educado lo mejor posible, y gracias a ellos soy hoy quien soy y he llegado hasta aquí, y muy especialmente a mis abuelos, que siempre han tratado de darme todo lo que a estado a su alcance y han hecho derroche de amor, cariño y cuidado.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Yoandy Rey Díaz Nápoles	Dra. Maritza Rodríguez Gual
Firma del Autor.	Firma del Tutor.
_____	_____
Dr. Joaquín González González	Ing. Yahima Vigo Valdez
Firma del Tutor.	Firma del Tutor.

Resumen

En la actualidad, no existe un programa comercial que cree el fichero de entrada al código MCNPX (del inglés, Monte Carlo N-Particle X) a partir de imágenes médicas multimodales (SPECT-CT) corregistradas en formato DICOM (del ingles, Digital Imaging and Communications in Medicine), necesario para el cálculo dosimétrico en pacientes con lesiones tumorales. De ahí, la necesidad de crear uno para Cuba.

El objetivo de este trabajo de tesis lo constituye: la creación de un software de interfaz para crear un fichero de entrada al código de Monte Carlo MCNPX a partir de imágenes médicas multimodales (SPECT-CT) corregistradas en formato DICOM.

Como resultado del trabajo se construyó e implementó el software de interfaz que permitirá al código de MCNPX calcular de forma más exacta la distribución de dosis paciente-específico, en los tratamientos de tumores con radionúclidos, contribuyendo a disminuir el segundo renglón de muerte en Cuba, el cáncer.

A partir de la lectura del fichero de entrada, creado por el software desarrollado en este trabajo de tesis, por el código MCNPX y su ejecución de forma satisfactoria, quedó demostrada la validez práctica y aplicabilidad de dicho software creado.

En este documento se recogen los resultados del estudio realizado a varios sistemas encargados del procesamiento de imágenes para la elaboración de un fichero de entrada para el código MCNPX. Se incluyen conceptos relacionados con la Medicina Nuclear y la Imagenología. Además, se muestran los resultados del diseño e implementación de la propuesta del sistema y se dejan algunas recomendaciones para próximas versiones del mismo.

Palabras claves
DICOM, SPECT, CT, MCNPX, imagen.

Índice

Introducción

En Cuba se invierten muchos recursos en la Salud Pública para aumentar la calidad de vida del pueblo. La segunda causa de muerte en el país es el cáncer, con tendencia a ser la primera en los próximos años.

Para el tratamiento del cáncer empleando radiofármacos, es preciso conocer la distribución de dosis en el tumor y los órganos a riesgo. Para esto en la actualidad se utilizan los métodos de cálculo dosimétrico basados en códigos Monte Carlo, por ser los más exactos, ya que tiene en cuenta heterogeneidades de tejidos de diferentes densidades, ej.: huesos y tejido blando.

El código MCNPX calcula la distribución de dosis en el órgano o tejido del paciente enfermo del cáncer, a partir de un fichero de entrada que contiene los datos de imágenes médicas de réplicas del cuerpo humano. Lo anterior dificulta la exactitud del cálculo de la distribución de dosis paciente-específico, en los tratamientos de tumores con radionúclidos.

Existen diferentes formatos de imágenes médicas (Analize, Interfile, DICOM), pero el DICOM (en inglés, Digital Imaging and Communications in Medicine) es con el que se están comercializando en la actualidad la mayoría del equipamiento médico. El formato DICOM es un estándar en comunicaciones de imágenes en medicina, que facilita el manejo de información médica entre hospitales y centros de investigación. Con el estándar DICOM se pueden adquirir imágenes de equipos de diferentes modalidades. Por tal motivo, en el presente trabajo de tesis se trabaja con el formato DICOM.

Existen evidencias de que el corregistro y fusión de imágenes de diferentes modalidades conlleva a una mayor exactitud diagnóstica. Por tal motivo, se trabaja con las imágenes corregistradas de diferentes modalidades.

Existen dos modalidades de imágenes médicas: las anatómicas (CT, MRI) y las funcionales (PET, SPECT).

Estas modalidades de imágenes médicas están dadas por la metodología utilizada para realizar la toma de las imágenes, así como la forma en la cual son representadas. Las modalidades más utilizadas en la medicina nuclear son: Tomografía por Emisión de Positrones (en inglés, Positron Emision Tomography o PET), Tomografía por Emisión de Fotón Único (en inglés, Single-Photon Emission Computed Tomography o

SPECT) y en la radioterapia: la Tomografía Computarizada (en inglés, Computed Tomography, CT) y la Resonancia Magnética (en inglés, Magnetic Resonance Image, MRI).

En la investigación se utiliza como imagen de medicina nuclear la SPECT y como imagen de radioterapia la CT. Las imágenes anatómicas ofrecen una excelente información estructural o anatómica, pero ofrecen menos información sobre las funciones que realizan las regiones visualizadas, porque no muestran los detalles funcionales. Las imágenes funcionales son métodos precisos para detectar el cáncer y las anomalías relacionadas con el metabolismo o funcionalidad, pero ellas no ofrecen los datos necesarios para localizar las lesiones con precisión.

Cada modalidad genera una información visual diferente de un mismo tejido. Lo que permite que el especialista tenga diferentes informaciones de un mismo tejido mediante el complemento de las imágenes.

Hoy en día, no existe ningún programa comercial que cree el fichero de entrada al código MCNPX, a partir de imágenes médicas multimodales (SPECT-CT) corregistradas en formato DICOM, para que el cálculo de dosis realizado por Monte Carlo sea más exacto.

De ahí, que el **problema científico** que aborda el presente trabajo de diploma sea:

Necesidad de obtener datos de las imágenes de pacientes reales para realizar el cálculo dosimétrico con el código MCNPX de forma más exacta.

Identificando como **Objeto de Estudio** del presente trabajo: programas de interfaz para obtener los datos de imágenes médicas y como **Campo de Acción:** los programas de interfaz entre las imágenes médicas multimodales (SPECT-CT) corregistradas en formato DICOM y el código Monte Carlo MCNPX.

El **Objetivo General** que se plantea en esta investigación**:**

Desarrollar un programa de interfaz, entre las imágenes médicas multimodales (SPECT-CT) corregistradas en formato DICOM y el código Monte Carlo MCNPX, que permita generar un fichero de entrada para que sea analizado por dicho código para realizar cálculos de distribución de dosis.

Con éstos propósitos se definen como **objetivos específicos** de ésta tesis:

1. Realizar un estudio a partir de la literatura especializada del marco teórico y las tecnologías y herramientas a utilizar en el desarrollo del software.
2. Realizar una propuesta del sistema a desarrollar usando sus características para su posterior implementación.
3. Realizar el diseño, implementación y validación del software aplicando las tecnologías y herramientas seleccionadas.
4. Realizar el estudio de factibilidad.

Este trabajo parte de la **hipótesis** de que: “Desarrollando un programa de interfaz entre las imágenes adquiridas de estudios de SPECT-CT corregistradas y el código Monte Carlo MCNPX será posible determinar la distribución de dosis absorbida en tumores y tejidos reales a partir de las imágenes de paciente específico”.

Aporte social de la tesis:

1. Disponer en el país de un programa de interfaz para cálculos dosimétricos mediante el código MCNPX basado en Monte Carlo a partir de imágenes corregistradas de SPECT-CT adquiridas en formato DICOM que permitirá determinar la distribución en 3D de dosis absorbida en tumores y tejidos normales a partir de las imágenes de paciente específico; para disminuir el índice de muertes por cáncer.

El presente trabajo de tesis se desarrolla en el marco de un Programa Ramal Nuclear (PRN/1-3/8–2010) titulado: Cálculo de dosis absorbida con MCNPX en pacientes a partir de imágenes médicas corregistradas (SPECT-CT) del Instituto Superior de Tecnologías y Ciencias Aplicadas (InSTEC) en el que participan físicos médicos del Instituto Nacional de Oncología y Radiobiología(INOR) de Cuba

El presente estudio está estructurado en 4 capítulos los cuales son descritos:

- Capítulo 1: Fundamentación Teórica.
  Se profundizan en los conceptos fundamentales para el desarrollo de una aplicación que sirva de interfaz para la entrada de datos al MCNPX. También se exponen las principales características de las tecnologías, herramientas, lenguajes y técnicas de desarrollo propuestas para la solución del problema actual, así como la justificación de la selección de las mismas.
- Capítulo 2: Características del Sistema.
  Se elabora una propuesta del sistema a implementar, se abarcan las características del sistema para un mejor entendimiento, detallando además

las historias de usuarios que proporcionan un mejor punto de vista acerca de lo que el cliente desea.

- Capítulo 3: Diseño, Implementación y Pruebas.
  En este capítulo se detalla la construcción del software, además se exponen las tareas generadas por cada historia de usuario (HU), así como las pruebas efectuadas sobre el mismo.
- Capítulo 4: Estudio de la factibilidad.
  Se realiza una estimación en tiempo y costo del proyecto, indicando si es factible o no realizar el programa.

Capítulo I: Fundamentación teórica

Introducción

A lo largo del desarrollo de este capítulo se profundizan algunos conceptos fundamentales para el desarrollo de una aplicación que sirva de interfaz para la entrada de datos al MCNPX. También se exponen las principales características de las tecnologías, herramientas, lenguajes y técnicas de desarrollo propuestas para la solución del problema actual, así como la justificación de la selección de las mismas.

1.1 Conceptos fundamentales

DICOM es un estándar con el cual se puede adquirir imágenes de equipos de diferentes modalidades. En el encabezado se almacenan la información sobre el paciente y la imagen médica como: nombre, sexo, parte del cuerpo que se analiza, fecha en que se realiza el análisis, así como, la información de los pixeles que permiten construir la imagen, el tamaño del voxel, el número de cortes, etc.

SPECT es una modalidad de imagen DICOM, de tipo funcional, la cual es computarizada por emisiones de fotones simples, esta permite ver detalles de la actividad en el tejido, muestra la funcionalidad o metabolismo de un órgano. En este caso, se administra al paciente por vía intravenosa, un isótopo radiactivo cuyas emisiones son recogidas por cámaras gammas.

CT es una modalidad de imagen DICOM, de tipo anatómica, esta permite ver detalles precisos de la anatomía del tejido, muestra imágenes detalladas de cualquier parte del cuerpo, incluyendo los huesos, los músculos, la grasa y los órganos. En este caso, el paciente se expone a radiación externa desde un tomógrafo

1.2 Estado del arte.

Diferentes grupos de investigadores en el mundo han desarrollado software para realizar cálculos dosimétricos empleando diferentes códigos Monte Carlo. Entre estos desarrollos se destacan por sus aplicaciones en condiciones de rutina clínica los siguientes:

- RMDP [1]
- 3D-ID/3D-RD [2]

- RTDS [3]
- DOSE3D [4]
- SIMDOS [5]
- SCMS [6]

Estos softwares han desarrollado diferentes interfaces entre el código Monte Carlo y las imágenes médicas de SPECT/CT para solucionar la implementación de cálculos dosimétricos en condiciones de rutina clínica empleando técnicas de Monte Carlo. El principal problema es que estas aplicaciones no están disponibles comercialmente, lo cual limita su difusión en el ámbito clínico. Esto ha conllevado a la necesidad de buscar soluciones nacionales que permitan introducir las técnicas de Monte Carlo para cálculos dosimétricos en pacientes en nuestro país.

En 2003 como parte de esta necesidad se desarrolla un software de interfaz titulado IMAGON3D [7] en el Instituto Superior de Tecnologías y Ciencias Aplicadas (INSTEC) en colaboración con el Instituto de Oncología y Radiobiología (INOR). Este software tiene la limitación de que sólo emplea imágenes SPECT en formato Interfile y se desarrolló con el propósito de comparar cálculos dosimétricos obtenidos para medios homogéneos, es decir tejidos de un mismo tipo, empleando el método de Monte Carlo y método de los factores S a nivel de voxel.

Otros centros del país como en Centro de Investigaciones Clínicas (CIC) en colaboración con el Centro de Protección e Higiene de la Radiaciones (CPHR) están destinando esfuerzos para desarrollar dosimetría paciente-específica empleando el código Monte Carlo MCNP5 [8].

Este trabajo de tesis se enmarca en la necesidad de incorporar no solo las imágenes SPECT sino también las imágenes de CT con el objetivo de tener en cuenta la distribución de densidades de los tejidos en cada paciente en los cálculos dosimétricos teniendo en cuenta el estándar DICOM para el intercambio de imágenes médicas.

Esta tesis representa un importante esfuerzo científico en el desarrollo de métodos que contribuyan a lograr que la terapia del cáncer con radiofármacos emisores beta sea una alternativa eficaz y segura para los pacientes oncológicos en nuestro país.

Con la simulación Monte Carlo y las imágenes de paciente-específico se puede reducir las incertidumbres en la estimación de las dosis absorbidas. Este es el impacto sustancial de este trabajo de tesis.

1.3 Lenguajes de programación

En la actualidad existen diversos lenguajes de programación tales como: C, C++, C#, Microsoft Visual Basic, Microsoft Visual Basic.NET y Java, entre otros. En ocasiones resulta complejo seleccionar el lenguaje de programación a utilizar para el desarrollo de un proyecto de software determinado. La razón es que elegir un lenguaje de programación depende de muchos factores como lo son: el tipo de programa que se desea realizar, la plataforma requerida para estos programas, incluso siendo poco objetivos, también entra el gusto por un lenguaje en específico. Una buena elección debería basarse en las fortalezas y debilidades de cada lenguaje para desarrollar una determinada tarea. El problema a resolver debería determinar el lenguaje de programación a utilizar.

1.3.1 Lenguaje de programación C / C++

Los lenguajes de programación C / C++ tienen un soporte muy fuerte en cualquier plataforma o sistema operativo que el programador utilice. Un programa escrito en C o C++ podrá ser copiado sin dificultad para otra computadora con un entorno diferente, y luego, podrá ser ejecutado con solo algunos cambios, por lo que brindan la posibilidad de escribir los programas solo una vez, y luego modificarlos para que sean capaces de correr en sistemas operativos diferentes.

C / C++ brindan el poder y la flexibilidad de manipular la memoria y acceder directamente al hardware. Esto no solo incrementa la posibilidad de que un error se encuentre alojado en su código, sino que incrementa también la cantidad de tiempo necesaria para depurar el programa con el objetivo de lograr su correcto funcionamiento.[9]

Estos lenguajes, han ido actualizándose lentamente a lo largo de los años pero, en la mayoría de los casos, no ofrecen soluciones integrales para las necesidades actuales, sino tan solo parches para ir resolviéndolas de manera puntual.[10] La mayor desventaja que presentan estos lenguajes es la dificultad en su aprendizaje, obligan a hacerlo todo manualmente.

1.3.2 Microsoft Visual Basic / Microsoft Visual Basic.NET

Visual Basic es uno de los lenguajes de programación más populares para el desarrollo de software. Un programador puede aprender Visual Basic y comenzar a utilizarlo mucho más rápido que si eligiera otro lenguaje de programación.

Visual Basic es fácil de aprender debido a que aísla al programador de los detalles técnicos de la programación de una computadora, a pesar de esto, los programadores

profesionales frecuentemente descartan Visual Basic y lo califican como un lenguaje “juguete”, debido a sus limitaciones a la hora de ser utilizado para el desarrollo de aplicaciones.[11]

Su mayor desventaja es en cuanto a su portabilidad, un programa escrito en Visual Basic solamente funciona sobre la familia de sistemas operativos Windows, nunca funcionará en otro tipo de sistema operativo.

Visual Basic .NET (VB.NET) es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET. Permite a los desarrolladores centrar el diseño en Windows, el Web y dispositivos móviles. Su introducción resultó muy controvertida, ya que debido a cambios significativos en el lenguaje VB.NET no es compatible hacia atrás con Visual Basic.

1.3.3 Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Java es un lenguaje de programación completamente portable, por lo que un programa escrito en Java podrá teóricamente correr en cualquier computadora o sistema operativo sin que sea necesaria modificación alguna [12]. Java utiliza un mecanismo llamado recolector automático de basura (automatic garbage collector) para eliminar los objetos creados por el programador en la memoria de la computadora. Esto trae consigo que no existan desbordes de memoria como en C / C++, y que el código sea más limpio y fácil de escribir. La recolección de basura de Java es un proceso prácticamente invisible al desarrollador. A pesar de sus grandes ventajas, Java también cuenta con un número de desventajas que deberán ser analizadas en el momento de tomar una decisión. Los programas en Java tienden a ser mucho más lentos que sus equivalentes escritos en otros lenguajes de programación. Además, Java es un lenguaje con una curva de aprendizaje bastante elevada.

1.3.4 C#

C# es un nuevo lenguaje propuesto por Microsoft para satisfacer las necesidades actuales y de un futuro cercano. C# es, por tanto, una herramienta, como todos los lenguajes de programación, pero adaptada al trabajo actual.

El objetivo de Microsoft ha sido la creación del primer lenguaje orientado a componentes, al estilo de Visual Basic, pero con la flexibilidad y potencia de C++ y sin muchas de sus complejidades. C# ha sido diseñado para una plataforma, la plataforma Microsoft .NET, en la que los servicios son ofrecidos en forma de componentes. Cuenta con construcciones sintácticas nativas para la definición, implementación y consumo de propiedades, métodos y eventos, lo cual le diferencia claramente de C++ o Java.

Para construir un componente con C# no es necesario hacer nada especial aparte de crear una nueva clase. Dicho de otro modo, cualquier clase de objeto C# es un componente, sin necesidad de crear GUID (Globally Unique Identifier) para interfaces y clases de componentes.

C# es un lenguaje completamente orientado a objetos con una gran cantidad de características y mejoras sobre sus predecesores. Al igual que Java, trabaja en un entorno manipulado de memoria, aislando al programador de los engorrosos detalles del hardware. A pesar de esto, el programador tendrá la posibilidad de renunciar a esta característica y lidiar por sí mismo con la manipulación de la memoria [13].

Se dice que su competir más cercano es Java, lenguaje con el que guarda un enorme parecido. En este aspecto, es importante señalar que C# incorpora muchos elementos de los que Java carece como: el rendimiento, el cual es mejor, soporta más tipos primitivos, incluyendo tipos numéricos sin signo, compilación condicional, aplicaciones multi-hilo simplificadas; y otros [14]. Las principales características que definen al lenguaje C# son:

- **Sencillez de uso:** C# elimina muchos elementos añadidos por otros lenguajes y que facilitan su uso y compresión, como por ejemplo ficheros de cabecera, o ficheros fuentes IDL.
- **Modernidad:** Al ser C# un lenguaje de última generación, incorpora elementos que se ha demostrado a lo largo del tiempo que son muy útiles para el programador, como tipos decimales o booleanos, así como una instrucción que permita recorrer colecciones con facilidad (instrucción foreach). Estos elementos hay que simularlos en otros lenguajes como C++ o Java.
- **Orientado a objetos:** C# como lenguaje de última generación es orientado a objetos. Además, C# soporta todas las características del paradigma de la programación orientada a objetos, como son la encapsulación, la herencia y el polimorfismo.
- **Orientado a componentes:** La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular.

- **Recolección de basura:** Todo lenguaje incluido en la plataforma .NET tiene a su disposición el recolector de basura.
- **Eficiente:** En C#, todo el código incluye numerosas restricciones para garantizar su seguridad, no permitiendo el uso de punteros.
- **Compatible:** Para facilitar la migración de programadores de C++ o Java a C#, no sólo se mantiene una sintaxis muy similar a la de los dos anteriores lenguajes, sino que también ofrece la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos, tales como las DLLs de la API de Win32.

Se decide utilizar el lenguaje de programación C#, de la plataforma .NET, para generar los artefactos de tipo software, pues C# es simple, eficaz, orientado a objetos, permite desarrollar aplicaciones rápidamente y mantiene la expresividad y elegancia de los lenguajes de tipo C; mientras que los demás lenguajes no poseen estas características a su mismo nivel.

1.4 Elección de la metodología

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas de metodologías que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas.

Como ejemplo de metodología tradicional contamos con Rational Unified Process(RUP) la cual se divide en 4 fases:

- Inicio, El Objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración, En esta etapa el objetivo es determinar la arquitectura óptima.
- Construcción, En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- Transición, El objetivo es llegar a obtener el proyecto.

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable al cliente. Esto trae

como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Como ejemplo de metodología ágil tenemos Extreme Programming (XP) la cual es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada para proyectos de corto plazo y corto equipo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

¿Qué propone XP?

- Empieza en pequeño y añade funcionalidad con retroalimentación continua
- El manejo del cambio se convierte en parte sustantiva del proceso
- El costo del cambio no depende de la fase o etapa
- No introduce funcionalidades antes que sean necesarias
- El cliente o el usuario se convierte en miembro del equipo

Por todo lo antes planteado se llega a la conclusión que la metodología más conveniente es XP ya que RUP no es recomendable para este trabajo de diploma pues es utilizado para realizar grandes proyectos donde el cliente no forme parte del equipo y este último este integrado por muchas personas y tenga una duración prolongada.

1.5 Herramientas a utilizar

1.5.1 Visual Studio 2010

.NET es la nueva tecnología desarrollada por Microsoft con los objetivos principales de mejorar los sistemas operativos y de obtener un entorno diseñado para el desarrollo y ejecución del software en forma de servicios que puedan ser accedidos a través de Internet de forma independiente al lenguaje de programación, sistema operativo y hardware utilizados tanto para desarrollarlos como para publicarlos.
Visual Studio .NET es la herramienta de desarrollo multilenguaje más completa para construir e integrar rápidamente aplicaciones y servicios Web XML (Extensible Markup Language, Lenguaje de Marcas Ampliable). Aumenta de un modo extraordinario la productividad de los desarrolladores y crea nuevas oportunidades de negocio. En su

diseño se han integrado a fondo los estándares y protocolos de Internet, como XML y SOAP (Simple Object Access Protocol), por lo que Visual Studio .NET simplifica considerablemente el ciclo de vida del desarrollo de aplicaciones.
Visual Studio 2010 tiene un elevado aumento de la productividad al escribir aplicaciones dirigidas a la nueva versión de .NET Framework, esto incluye la ampliación de tipos de proyectos, la reducción de tareas mundanas y siempre en evolución y los aspectos de equipo orientados a la ingeniería de software.

A las mejoras de desempeño, escalabilidad y seguridad con respecto a la versión anterior, se agregan entre otras, las siguientes novedades:

- **Mejora en las capacidades de Pruebas Unitarias:** son ejecutas más rápido independientemente de si lo hacen en el entorno IDE o desde la línea de comandos.
- **Visual Studio Tools for Office (VSTO):** integrado con Visual Studio 2010 es posible desarrollar rápidamente aplicaciones de alta calidad basadas en la interfaz de usuario de Office que personalicen la experiencia del usuario y mejoren su productividad en el uso de Word, Excel, PowerPoint, Outlook, Visio, InfoPath y Project.
- **LINQ (Language Integrated Query):** conjunto de herramientas diseñado para reducir la complejidad del acceso a Base de Datos.
- **Soluciones multiplataforma:** Visual Studio 2010 permite la creación de soluciones multiplataforma adaptadas para funcionar con las diferentes versiones de .Net Framework: 2.0. (Incluido con Visual Studio 2005), 3.0 (incluido en Windows Vista), 3.5 (incluido con Visual Studio 2008) y 4.0 (incluido con Visual Studio 2010 y Windows 7).
- **WPF (Windows Presentation Fundation):** Permite desarrollar aplicaciones con un estándar de programación con una curva de aprendizaje muy baja para los programadores web, revolucionando la programación de interfaces de escritorio, con una alta calidad de interfaz de usuario muy personalizable, permitiendo aplicarle estilos a todos los componentes.

Se decide utilizar Visual Studio 2010 como IDE de desarrollo ya que ofrece la visión de las aplicaciones clientes inteligentes, al permitir a los desarrolladores con avanzadas herramientas de desarrollo, y otras características innovadoras, la creación de aplicaciones de manera rápida a través de diversas plataformas.

1.6 Resultados esperados

Obtener un software que sirva de interfaz al código de MCNPX para realizar cálculos dosimétricos con gran exactitud para el tratamiento de pacientes a partir de datos reales de estos y no de réplicas del cuerpo humano.

1.7 Conclusiones

El estudio del formato DICOM permitió la selección de las técnicas y algoritmos a utilizar en esta investigación, ya que es un estándar moderno de imágenes y por lo tanto lleva un tratamiento diferente, al no presentar la misma estructura que las imágenes convencionales; además quedó evidenciada la necesidad de implementar este software. Se establecieron la metodología y herramientas para el modelado y desarrollo del mismo según sus características.

Capítulo II: Características del sistema

Introducción

En el presente capítulo se especifican las características de la propuesta de solución del sistema. Se realiza una descripción de la problemática a resolver y se elabora una propuesta del sistema a implementar detallando las Historias de Usuarios (HU) propuestas por la metodología de desarrollo XP. Además, se realiza la estimación del esfuerzo por cada una de las HU identificadas como base para elaborar el plan de iteraciones y obtener el plan de entrega.

2.1 Propuesta del sistema

Hoy en día no existe una herramienta gratis o de fácil acceso económico para nuestro país con la cual se cree a partir de imágenes corregistradas SPECT-CT en formato DICOM un fichero de entrada para el código MCNPX para el cálculo dosimétrico de pacientes-específicos para el tratamiento de cáncer.

Para resolver esta problemática se propone realizar un software de escritorio para que permita cargar imágenes DICOM, visualizar sus etiquetas, convertir a otros formatos la imagen y crear y salvar un fichero de entrada para el código MCNPX el cual realizaría el cálculo dosimétrico.

Para confeccionar el fichero de entrada al MCNPX es necesario contar con el mapa de la distribución de actividad que se obtendrá a partir de la imagen SPECT y también con el mapa de la distribución de densidad que se obtendrá a partir de la imagen CT del paciente.

2.2 Requisitos no funcionales del sistemas

Rendimiento

El sistema deberá estar en capacidad de prestar el servicio con unos niveles aceptables de desempeño, teniendo en cuenta que el tiempo de respuesta esperado para ejecución del sistema no debe exceder de 30 minutos.

Mantenibilidad

El sistema deberá contar con una especificación técnica de tal manera que un profesional de la rama de medicina pueda entender su funcionamiento y hacer los ajustes necesarios.

Explicará qué hace la aplicación, a nivel técnico, e indicará qué hacer ante la posible ocurrencia de errores comunes de orden técnico y cómo solucionarlos. Estos errores no son del sistema, sino errores que el usuario puede cometer. Explicará a los usuarios o clientes a nivel técnico, cómo se instala la aplicación y sus requerimientos.

Usabilidad

El sistema deberá tener una interfaz gráfica uniforme a través del mismo incluyendo interfaces, menús y opciones. El sistema deberá ser de uso intuitivo, de tal forma que se reduzca los tiempos de entrenamiento, soporte y prueba por parte del usuario.

Hardware

Para el funcionamiento de la aplicación se debe disponer de una computadora con microprocesador Pentium IV o superior, con 512 MB de RAM o superior, 100 MB de espacio libre en disco duro o superior.

Software

Se debe disponer para el uso de la aplicación, del Sistema Operativo Windows XP o superior. Se debe instalar el Net Framework v4.0.

2.3 Historias de usuario

Las historias de usuario son escritas por el cliente en su propio lenguaje. Describe lo que el sistema debe realizar, especificando detalles mínimos para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios.

A continuación se muestra la plantilla para realizar las Historias de Usuario con una breve descripción de cada parámetro para un mejor entendimiento:

Historia de Usuario	
Número: (Número de la historia de usuario incremental en el tiempo)	**Nombre de Historia de Usuario:** (El nombre de la historia de usuario sería para identificarlas mejor entre los desarrolladores y el cliente)

Modificación de Historia de Usuario Número: (si sufrió alguna modificación anterior)	
Usuario: (Involucrados en el desarrollo de la HU)	**Iteración Asignada:** (Numero de la iteración)
Prioridad en negocio: (Alta / Media / Baja)	**Puntos estimados**: (El tiempo estimado que se demorará el desarrollo de la HU)
Riesgo en Desarrollo: (Alta / Media / Baja)	**Puntos Reales:** (El tiempo que se demoró en realidad el desarrollo de la HU)
Descripción: (Breve descripción de la HU)	
Observaciones: Señalamiento o advertencia del sistema.	
Prototipo de interfaz:	

Tabla 1: Plantilla de HU

2.3.1 La prioridad en el negocio

Alta: Se le otorga a las Historias de Usuario que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.

Media: Se le otorga a las Historias de Usuario que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.

Baja: Se le otorga a las Historias de Usuario que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.

2.3.2 El riesgo en su desarrollo

Alto: Cuando en la implementación de las Historias de Usuario se consideran la posible existencia de errores que lleven la inoperatividad del código.

Medio: Cuando pueden aparecer errores en la implementación de la Historias de Usuario que puedan retrasar la entrega de la versión.

Bajo: Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto. El cliente y el equipo de desarrollo trabajan en conjunto para definir como agrupar las Historias de Usuario para su lanzamiento.

2.3.3 Historias de usuarios concebidas

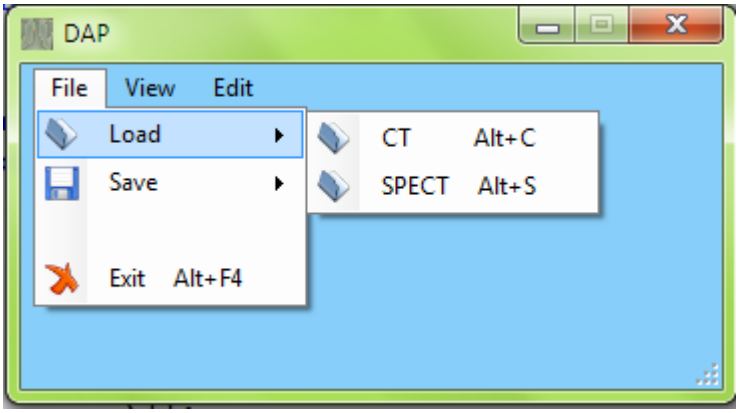
Historia de Usuario	
Número: 1	**Nombre de Historia de Usuario:** Cargar imagen
Modificación de Historia de Usuario Número:	
Usuario: Médico	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados**: 5
Riesgo en Desarrollo: Alta	**Puntos Reales:** 5
Descripción: Inicia cuando el usuario selecciona la opción que permite cargar la imagen que desea analizar. El sistema verifica que la imagen sea formato DICOM.	
Observaciones:	
Prototipo de interfaz: 	

Tabla 2: Historia de usuario Nro. 1: Cargar imagen CT.

Historia de Usuario	
Número: 2	**Nombre de Historia de Usuario:** Cargar imagen SPECT

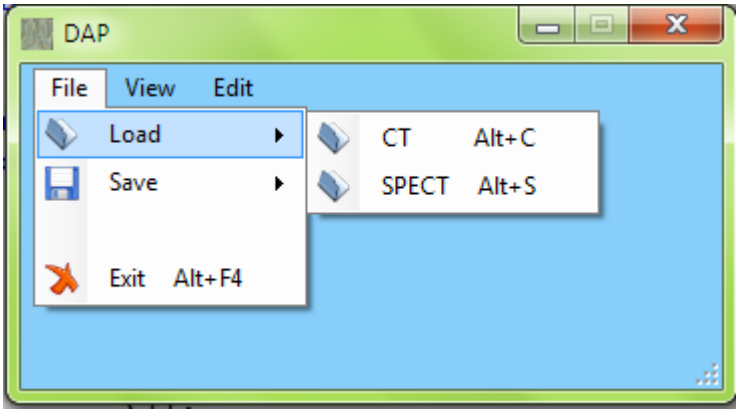
Modificación de Historia de Usuario Número:	
Usuario: Médico	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados**: 5
Riesgo en Desarrollo: Alta	**Puntos Reales:** 5
Descripción: Inicia cuando el usuario selecciona la opción que permite cargar la imagen que desea analizar. El sistema verifica que la imagen sea formato DICOM.	
Observaciones:	
Prototipo de interfaz: 	

Tabla 3: Historia de usuario Nro. 2: Cargar imagen SPECT.

Historia de Usuario	
Número: 3	**Nombre de Historia de Usuario:** Cambiar formato de imagen
Modificación de Historia de Usuario Número:	
Usuario: Médico	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados**: 5
Riesgo en Desarrollo: Alta	**Puntos Reales:** 6
Descripción: Inicia cuando el usuario selecciona la opción Salvar imagen escogiendo la modalidad de la imagen que desea salvar. Se muestra una interfaz que permite escoger la ubicación y extensión de la imagen, así como especificar su nombre. El usuario introduce los datos y se procede a guardar la imagen	

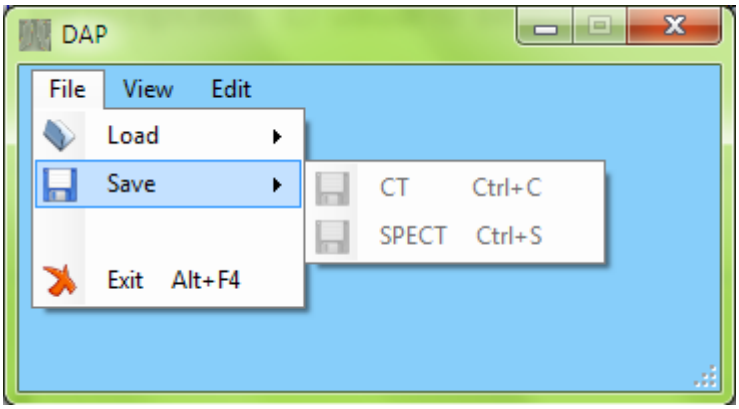
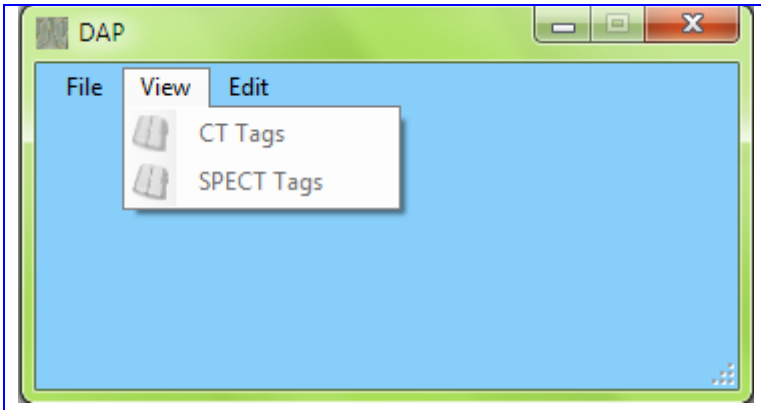
Observaciones:
Prototipo de interfaz: 

Tabla 4: Historia de usuario Nro. 3: Cambiar formato de imagen.

Historia de Usuario	
Número: 4	**Nombre de Historia de Usuario:** Visualizar las etiquetas de la imagen
Modificación de Historia de Usuario Número:	
Usuario: Médico	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados**: 5
Riesgo en Desarrollo: Alta	**Puntos Reales:** 6
Descripción: Inicia cuando el usuario selecciona la opción visualizar etiquetas de la imagen escogiendo la modalidad de la imagen de la cual se desea ver las etiquetas. Se muestra una interfaz que permite ver los datos complementarios de la imagen.	
Observaciones: Los datos de las imágenes cambian en dependencia de la modalidad y/o del equipo de radioterapia.	
Prototipo de interfaz:	



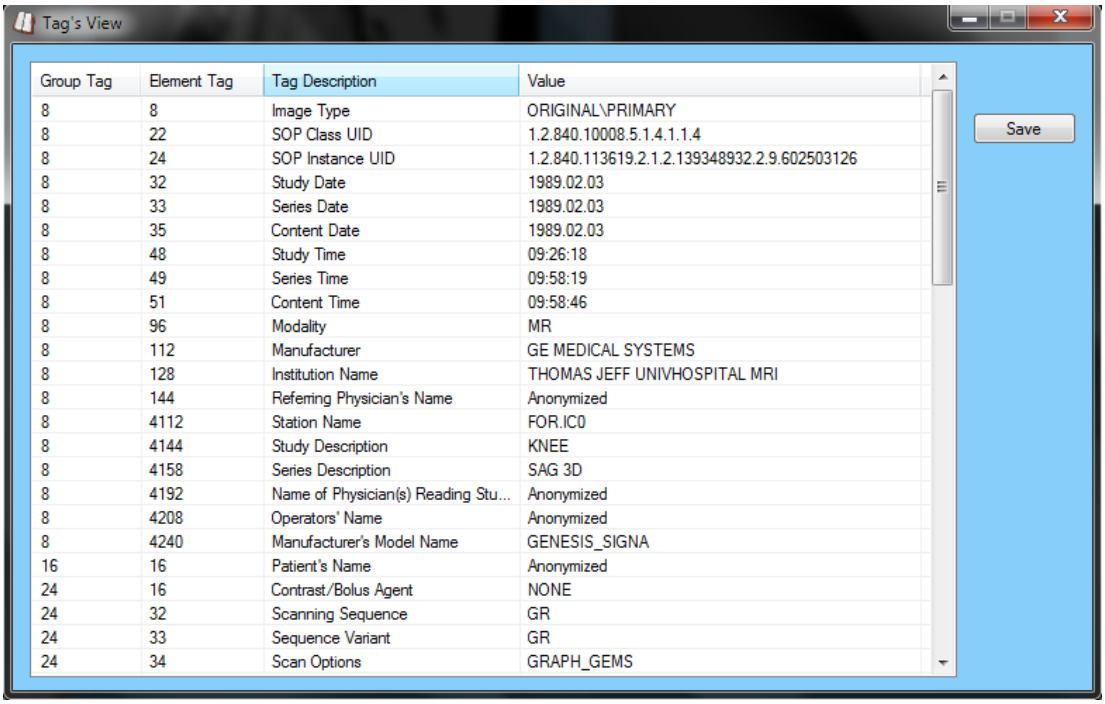


Tabla 5: Historia de usuario Nro. 4: Visualizar las etiquetas de la imagen.

Historia de Usuario	
Número: 5	**Nombre de Historia de Usuario:** Salvar las etiquetas de la imagen
Modificación de Historia de Usuario Número:	
Usuario: Médico	**Iteración Asignada:** 1
Prioridad en negocio: Baja	**Puntos estimados**: 4
Riesgo en Desarrollo: Medio	**Puntos Reales:** 4
Descripción: A partir de la historia de usuario 3 (**Visualizar las etiquetas de la**	

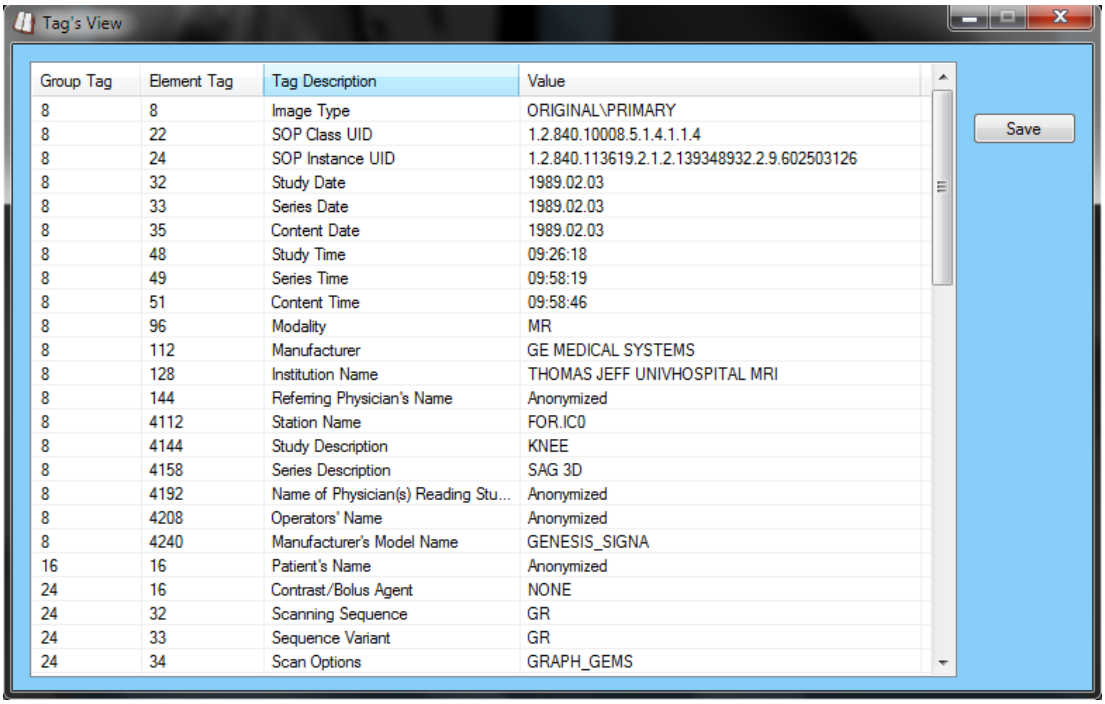
imagen), el usuario opta por salvar las etiquetas, seleccionando de una interfaz el formato, nombre y ubicación para proceder a salvar dicha información en un fichero de texto.
Observaciones:
Prototipo de interface: 

Tabla 6: Historia de usuario Nro. 5: Salvar las etiquetas de la imagen.

Historia de Usuario	
Número: 6	**Nombre de Historia de Usuario:** Crear fichero de entrada al código MCNPX
Modificación de Historia de Usuario Número:	
Usuario: Médico	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados**: 7
Riesgo en Desarrollo: Alto	**Puntos Reales:** 7
Descripción: Inicia cuando el usuario, una vez cargadas las imágenes de diferentes modalidades, selecciona crear el fichero de entrada al código MCNPX. Para esto se	

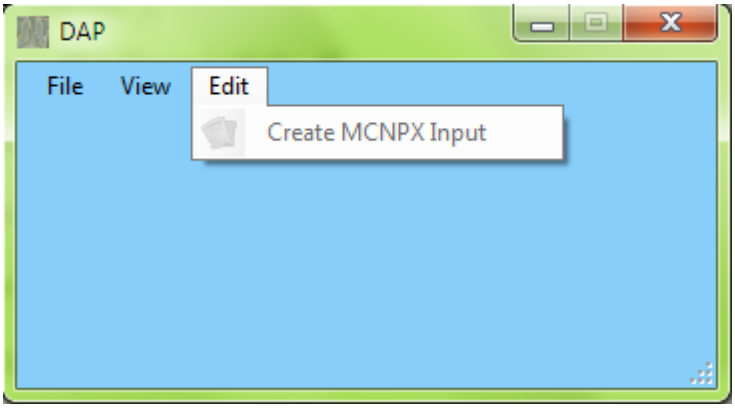
debe crear el mapa de densidad de la imagen CT, el mapa de actividad de la imagen SPECT y el sistema muestra una interfaz para introducir los siguientes datos: • **Radionúclido.** • **Radiation mode.** • **Source Type.** • **Sensibility.** El usuario procede a introducir los datos solicitados y se procede a realizar los cálculos. Luego se muestra una interfaz para seleccionar el formato, nombre y ubicación para salvar la información procesada en un fichero de texto.
Observaciones:
Prototipo de interface: 

Tabla 7: Historia de usuario Nro. 6: Crear fichero de entrada al código MCNPX.

2.4 Planificación

En la presente fase se lleva a cabo la estimación del esfuerzo que costará la implementación de cada historia de usuario, como en la metodología XP las métricas son libres, puede utilizarse cualquier criterio para medir el desempeño del desarrollo del software en cuestión. Una de las métricas más utilizadas en este tipo de metodología es la medida de puntos; un punto en esta métrica es el equivalente a una semana de trabajo, donde los miembros de los equipos de desarrollo trabajan sin interrupciones.

2.4.1 Estimación de esfuerzo por historias de usuarios.

Para el desarrollo de la aplicación propuesta se realizó una estimación de esfuerzo por cada una de las historias de usuario identificadas y los resultados obtenidos se muestran a continuación:

Historia de Usuario	Puntos de estimación
Cargar imagen CT	1
Cargar imagen SPECT	1
Cambiar formato de imagen	2
Visualizar las etiquetas de la imagen	2
Salvar las etiquetas de la imagen.	1
Crear fichero de entrada al código MCNPX.	3

Tabla 8: estimacion de esfuerzo por Historia de Usuario

2.4.2 Plan de Iteraciones.

Después de ser identificadas las historias de usuario y estimado el esfuerzo dedicado a la realización de cada una de ellas se procede a la realización de la planificación de la etapa de implementación del presente trabajo. Para un mejor desempeño en el desarrollo se ha establecido una división de la implementación en 2 iteraciones.

2.4.2.1 Iteración 1

En la iteración 1 se implementarán las historias de usuario 1, 2, 3, 4 y 5. Al finalizar esta iteración se contará con la primera versión de prueba de la aplicación donde se podrá observar los datos complementarios de las imágenes.

2.4.2.2 Iteración 2

En la iteración 2 será implementada la historia de usuario 6 la cual como las demás es prioritaria para la aplicación. Al terminar esta iteración se contará con una versión 1.0 del producto final y como resultado el software se pondrá en funcionamiento para evaluar su desempeño.

2.4.3 Plan de duración de las iteraciones.

El plan de duración de las iteraciones es el encargado de mostrar las historias de usuario que serán implementadas en cada una de las iteraciones, así como la duración estimada y el orden de implementación de cada una de ellas.

Iteración	Orden de la Historias de usuario a implementar.	Duración total
1	Cargar imagen CT Cargar imagen SPECT Salvar Imagen Visualizar las etiquetas de la imagen Salvar las etiquetas de la imagen	7 semanas
2	Crear fichero de entrada al código MCNPX	3 semanas

Tabla 9: Plan de duración de iteraciones

Aplicación	Final de la Iteración 1 (20 de Abril del 2011)	Final de la Iteración 2 (3 de Junio del 2011)
DAP	DAP v0.1	DAP v0.2

Tabla 10: Plan de entregas

2.5 Conclusiones

En este capítulo se han representado las principales características del sistema, en función de realizar un software capaz de cumplir con las expectativas y requisitos del cliente. A favor de dicho sistema se propuso la planificación del equipo de desarrollo, compuesta por iteraciones donde de forma incremental se implementará el sistema propuesto. Además, se obtuvo un listado de requisitos no funcionales que la aplicación debe cumplir. Con el resultado del desarrollo del capítulo, se da paso a la próxima fase de desarrollo del sistema.

Capítulo III: Diseño, implementación y pruebas

Introducción

La metodología XP propone que la implementación de un producto debe realizarse de forma iterativa. Después del desarrollo de cada iteración se obtiene un producto funcional que debe ser mostrado al cliente y previamente probado para incrementar la visión de los desarrolladores y clientes de posibles cambios.

En el presente capítulo se detalla la construcción de cada una de las iteraciones del software, además se exponen las tareas generadas por cada HU, así como las pruebas efectuadas sobre el mismo.

3.1 Patrones de Diseño

Patrones de diseño, del inglés Design Patterns, son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan. [15]

Los patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades) representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Los patrones de diseño pretenden:

• Proporcionar catálogos de elementos reusables en el diseño de sistemas software.

• Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.

• Formalizar un vocabulario común entre diseñadores.

• Estandarizar el modo en que se realiza el diseño.

• Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

3.1.1 Patrones básicos de asignación de responsabilidades

Los patrones empleados en el diseño del sistema propuesto fueron:

- **Bajo Acoplamiento:** Debe haber pocas dependencias entre las clases. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga

la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

- **Alta Cohesión:** La información que almacena una clase debe de ser coherente y debe estar relacionada con la clase. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos.

- **Experto:** Es el principio básico de asignación de responsabilidades. Indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene encapsulada.

- **Creador:** Ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos o clases, los cuales tendrán la información necesaria para realizar la creación del objeto.

3.2 Estándares de codificación

Un estándar de codificación es “Un conjunto de reglas de notación y nomenclatura, específicas de cada lenguaje de programación, que se usan y se siguen durante la fase de implementación (codificación) de una aplicación y reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores que no son detectados por los compiladores, reduciendo el tiempo y coste de las actividades de depuración y pruebas necesarias para la detección y corrección de los mismos.” (10) Las convenciones de nombres hacen los programas más entendibles haciéndolos más fácil de leer. También pueden dar información sobre la función de un identificador, por ejemplo, cuando es una constante, un paquete, o una clase, que puede ser útil para entender el código. En la siguiente tabla se especifican las convenciones de nombres utilizadas

Tipos de identificadores	**Reglas para nombres**	**Ejemplos**
Clases	Los nombres de las clases cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de las clases simples y descriptivos.	class Usuario class MaterialPobre

Métodos	Cuando son compuestos tendrán la primera letra en mayúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.	Almacenar(); AlmacenarEnvase();
Variable	Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Los nombres de variables no deben empezar con los caracteres subguión "_".Los nombres de variables de un solo caracter se deben evitar, excepto para variables índices temporales.	Int a; var j; string cadena;

Constantes	Los nombres de las variables declaradas como constantes deben ir totalmente en mayúsculas, siempre y cuando no sea una palabra completa.	const int IN = 30;
		const char CP = 'P';
		const int pixel = 15;

3.3 Tarjetas Clase – Responsabilidad – Colaborador

Una de las tareas más importantes en el diseño de una aplicación la constituye el definir correctamente las clases que contendrán la lógica del negocio. La metodología XP no obliga a la realización de diagramas UML para la presentación de las mismas y en su lugar propone el uso de tarjetas CRC (Clase-Responsabilidad-Colaboración). Estas ayudan desenfocar el aspecto procedimental destacando así la orientación a objetos.

El nombre de la clase está presente en forma de título en la tarjeta, en la izquierda las funcionalidades (responsabilidades) y a la derecha las clases que se implican en cada funcionalidad (colaboración).

A continuación se describen las tarjetas CRC pertenecientes al diseño de la aplicación, definiendo y simulando los escenarios que garantizan el buen funcionamiento del diseño.

Clase: Edit	
Responsabilidad	**Colaboración**
Cargar DICOM Salvar DICOM Crear Archivo de entrada MCNPX	Calc

Tabla 11: Tarjeta CRC Clase: Edit

Clase: Calc	
Responsabilidad	**Colaboración**
Localizar Voxels Calcular Densidad Calcular Actividad	Tpoint

Tabla 12: Tarjeta CRC Clase: Calc

Clase: Tpoint	
Responsabilidad	**Colaboración**
Almacenar posición de Voxels	

Tabla 13: Tarjeta CRC Clase: Tpoint

3.4 Tareas de Ingeniería

Cada una de estas historias de usuario se transformará en tareas que serán desarrolladas por programadores, dentro del equipo de desarrollo, aplicando la práctica de la Programación en parejas. Cada tarea de desarrollo corresponderá a un periodo de uno a tres días de desarrollo.

Historia de Usuario	**Tarea de Ingeniería**
Cargar imagen CT	Cargar imagen
Cargar imagen SPECT	Cargar imagen
Salvar Imagen	Salvar Imagen
Visualizar las etiquetas de la imagen	Visualizar las etiquetas de la imagen
Salvar las etiquetas de la imagen	Salvar las etiquetas de la imagen
Crear fichero de entrada al código MCNPX	Crear mapa de densidad Crear mapa de actividad Crear fichero

Tabla 14: Distribución de las tareas por cada historia de usuario.

Tareas detalladas

Tarea de Ingeniería	
Número Tarea: [Los números deben ser consecutivos]	**Número Historia de Usuario:** [Número de la historia de usuario a la que pertenece la tarea]
Nombre Tarea: [Nombre que identifica a la tarea.]	
Tipo de Tarea : [Las tareas pueden ser de: Desarrollo, Corrección, Mejora, Otra(Especificar)]	**Puntos Estimados:** [Tiempo en semanas que se le asignará. (Estimado)]
Programador Responsable: [Nombre y Apellidos del programador]	
Descripción: [Breve descripción de la tarea.]	

Tabla 15: Plantilla de Tarea de Ingeniería

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1 y 2
Nombre Tarea: Cargar Imagen CT.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Yoandy Rey Diaz Napoles	
Descripción: El usuario selecciona la imagen que quiere analizar, el sistema confirma sus datos y ofrece la visualización de la imagen.	

Tabla 16: Tarea de Ingeniería Nro. 1: Cargar Imagen.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 3
Nombre Tarea: Salvar Imagen.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Yoandy Rey Diaz Napoles	

Descripción: El usuario selecciona la imagen que quiere salvar, el sistema confirma sus datos y salva de la imagen.

Tabla 17: Tarea de Ingeniería Nro. 2: Salvar Imagen.

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 4
Nombre Tarea: Visualizar las etiquetas de la imagen.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Yoandy Rey Diaz Napoles	
Descripción: El cliente selecciona la imagen de la cual quiere ver las etiquetas, el sistema abre un nuevo formulario mostrando las etiquetas de la imagen en cuestión.	

Tabla 18: Tarea de Ingeniería Nro. 3: Visualizar las etiquetas de la image.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 5
Nombre Tarea: Salvar las etiquetas de la imagen.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Yoandy Rey Diaz Napoles	
Descripción: A partir de la tarea de ingeniería 5 (Visualizar las etiquetas de la imagen CT), el usuario opta por salvar las etiquetas, seleccionando de un cuadro de dialogo de salva de archivos el formato, nombre y ruta donde el sistema procede a salvar dicha información en un fichero de texto.	

Tabla 19: Tarea de Ingeniería Nro. 4: Salvar las etiquetas de la imagen.

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 6
Nombre Tarea: Crear mapa de densidad	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1

Programador Responsable: Yoandy Rey Diaz Napoles
Descripción: se debe convertir la información de los pixel de hexadecimal a decimal, para su tratamiento, luego se procede a verificar según una tabla de materiales, en que rango se encuentra el pixel para asociarlo con ese material, este proceso se debe repetir para cada pixel por separado y luego debe ser cuantificado.

Tabla 20: Tarea de Ingeniería Nro. 5: Crear mapa de densidad.

Tarea de Ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** 6
Nombre Tarea: Crear mapa de actividad	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Yoandy Rey Diaz Napoles	
Descripción: se debe convertir la información de los pixel de hexadecimal a decimal, para su tratamiento, se le solicitara al usuario que introduzca datos como: • Radionúclido. • Radiation mode. • Source Type. • Sensibility. , luego se procede a aplicar la fórmula de actividad (actividad= conteos por segundo / sensibilidad), este proceso se debe repetir para cada pixel por separado.En dependencia de los datos entrados por el usuario de conforma con un tipo de información o otra el resto del mapa de actividad	

Tabla 21: Tarea de ingeniería Nro 6: Crear mapa de actividad

Tarea de Ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** 6
Nombre Tarea: Crear fichero	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Yoandy Rey Diaz Napoles	
Descripción: una ves creado los mapas de densidad y actividad, se procede a crear un fichero escribiendo en el todos los datos adquiridos con las siguientes normas: • No se debe exceder una línea de 85 caracteres. • Los primeros cinco caracteres de una línea deben ser ”espacio”	

- Los comentarios deben tener un carácter ‘c’ delante para especificar que es una línea de comentarios

3.5 Pruebas

Las pruebas son procesos que permiten verificar y revelar la calidad de un producto software, siendo uno de los pilares de la metodología XP. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa o software. Básicamente es una fase en el desarrollo del software.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

3.5.1 Pruebas unitarias

Las pruebas unitarias permiten probar el correcto funcionamiento del código de un componente para asegurarse del correcto funcionamiento de cada uno por separado. Luego, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión. El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas.

Las unidades de test o pruebas constituyen unos de los pilares básicos de la Extreme Programming (XP). Uno de los errores que se suele cometer es pensar que podemos dejar la construcción de los test para los últimos meses en la realización de un proyecto. Descubrir todos los errores que pueden aparecer lleva tiempo, y más si se deja la depuración de todos para el final.

Las unidades de test están directamente relacionadas con el concepto de posesión del código. En cierta manera, una parte del código no será reemplazado si no supera los test que existen para ese código.

Después de cada modificación, se pueden emplear los test para verificar que un cambio en la estructura no introduce un cambio en la funcionalidad. Sin embargo, si se añaden nuevas capacidades al código, se tiene que rediseñar la unidad de test, para adaptarse a la nueva funcionalidad, de esta manera, la probabilidad de que exista un fallo en ambos (test y código) es menor.

Si se crean los test después de la creación del código habría que hacerlos utilizando ese código como un generador, se reubican los fallos del uno al otro. De aquí la importancia de la creación de las unidades de prueba antes que el código, para que sea independiente de este

Las pruebas se convierten en una herramienta de desarrollo, no es un paso de verificación que puede despreciarse si a uno le parece que el código está bien.

A continuación se muestran fragmentos de código y resultados de las pruebas unitarias realizadas.

```
[TestClass()]
public class CalcTest
{


    private TestContext testContextInstance;

    /// <summary>
    ///Gets or sets the test context which provides
    ///information about and functionality for the current test run.
    ///</summary>
    public TestContext TestContext
    {
        get
        {
            return testContextInstance;
        }
        set
        {
            testContextInstance = value;
        }
    }
```

Ilustración 1: Código de la clase a probar

```
    /// <summary>
    ///A test for HextoDec
    ///</summary>
    [TestMethod()]
    [DeploymentItem("DAP.exe")]
    public void HextoDecTest()
    {
        try
        {
             string num = "400"; // TODO: Initialize to an appropriate value
        int expected = 1024; // TODO: Initialize to an appropriate value
        int actual;
        actual = Calc.HextoDec(num);
        Assert.AreEqual(expected, actual);
        }
        catch (Exception e)
        {
            Assert.Inconclusive("Verify the correctness of this test method."+e.Message);
        }

    }
```

Ilustración 2: Código de la prueba unitaria 2

```
        /// <summary>
        ///A test for VoxelNumber
        ///</summary>
        [TestMethod()]
        [DeploymentItem("DAP.exe")]
        public void VoxelNumberTest()
        {
            try
            {
                int slices = 100; // TODO: Initialize to an appropriate value
                int w = 3; // TODO: Initialize to an appropriate value
                int h = 3; // TODO: Initialize to an appropriate value
                int expected = 900; // TODO: Initialize to an appropriate value
                int actual;
                actual = Calc.VoxelNumber(slices, w, h);
                Assert.AreEqual(expected, actual);
            }
            catch (Exception e)
            {

                Assert.Fail("Verify the correctness of this test method." + e.Message);
            }


        }
```

Ilustración 3: Código de la prueba unitaria 2

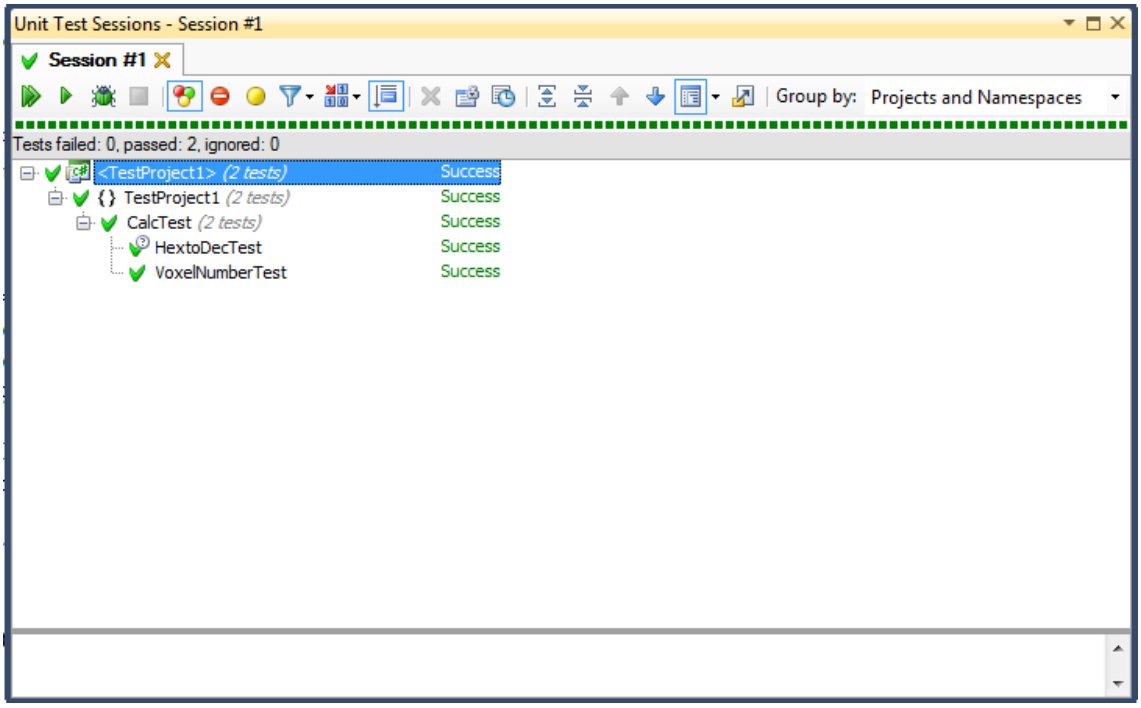


Ilustración 4: Pruebas unitarias

3.5.2 Pruebas de Aceptación

El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento.

Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a testear cuando una historia de usuario ha sido correctamente implementada.

Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento y no se considera completa hasta que no supera sus pruebas de aceptación. Esto significa que debe desarrollarse un nuevo test de aceptación para cada iteración o se considerará que el equipo de desarrollo no realiza ningún progreso.

Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas.

La garantía de calidad es una parte esencial en el proceso de XP. La realización de este tipo de pruebas y la publicación de los resultados debe ser los más rápido posibles, para que los desarrolladores puedan realizar con la mayor rapidez los cambios que sean necesarios.

Caso de Prueba Aceptación	
Código: 1	**Número Historia de Usuario:** 1
Nombre: Cargar imagen CT.	
Descripción: Prueba para verificar la funcionalidad Cargar imagen CT.	
Condiciones de Ejecución:	
Entrada/Pasos de ejecución: Seleccionar un archivo de formato DICOM	
Resultado Esperado: Se cargó correctamente la Imagen.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 22: Caso de Prueba Aceptación: Cargar imagen CT

Caso de Prueba Aceptación	
Código: 2	**Número Historia de Usuario:** 2
Nombre: Cargar imagen SPECT.	
Descripción: Prueba para verificar la funcionalidad Cargar imagen SPECT.	
Condiciones de Ejecución:	
Entrada/Pasos de ejecución: Seleccionar un archivo de formato DICOM	
Resultado Esperado: Se cargó correctamente la imagen.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 23: Caso de Prueba Aceptación: Cargar imagen SPECT.

Caso de Prueba Aceptación	
Código: 3	**Número Historia de Usuario:** 3
Nombre: Salvar imagen	
Descripción: Prueba para verificar la funcionalidad Salvar imagen.	
Condiciones de Ejecución: Se debe realizar una carga previa de la imagen a salvar	
Entrada/Pasos de ejecución: Seleccionar una dirección, nombre y extensión para realizar la salva.	
Resultado Esperado: Se salvó correctamente la Imagen.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 24: Caso de Prueba Aceptación: Salvar imagen

Caso de Prueba Aceptación	
Código: 5	**Número Historia de Usuario:** 4
Nombre: Visualizar las etiquetas de la imagen.	
Descripción: Prueba para verificar la funcionalidad Visualizar las etiquetas de la imagen.	
Condiciones de Ejecución: Se debe realizar una carga previa de la imagen, la cual contiene las etiquetas a visualizar	
Entrada/Pasos de ejecución: Seleccionar la opción ver etiquetas	
Resultado Esperado: Se visualizó correctamente las etiquetas la Imagen.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 25: Caso de Prueba Aceptación: Visualizar las etiquetas de la imagen

Caso de Prueba Aceptación	
Código: 7	**Número Historia de Usuario:** 5
Nombre: Salvar las etiquetas de la imagen.	
Descripción: Prueba para verificar la funcionalidad Salvar las etiquetas de la imagen.	
Condiciones de Ejecución: Se debe visualizar las etiquetas de la imagen	
Entrada/Pasos de ejecución: Seleccionar la opción ver etiquetas, luego se puede salvar las etiquetas mostradas	
Resultado Esperado: Se salvaron correctamente las etiquetas la Imagen.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 26: Caso de Prueba Aceptación: Salvar las etiquetas de la imagen.

Caso de Prueba Aceptación	
Código: 9	**Número Historia de Usuario:** 6
Nombre: Crear fichero de entrada al código MCNPX.	
Descripción: Prueba para verificar la funcionalidad Crear fichero de entrada al código MCNPX.	
Condiciones de Ejecución: Deben estar cargadas los dos tipos de imágenes, una CT y otra SPECT	
Entrada/Pasos de ejecución: Seleccionar la opción crear fichero de entrada, luego se procede a recolectar y procesar la información de las imágenes para ser guardadas en un fichero estándar.	
Resultado Esperado: Se creó correctamente el fichero de entrada al código MCNPX.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 27: Caso de Prueba Aceptación: Crear fichero de entrada al código MCNPX.

3.6 Conclusiones

En el capítulo se obtuvieron los artefactos propuestos por la metodología XP para la etapa de producción, demostrando el desarrollo satisfactorio del software en un período corto de tiempo. Se realizó una breve descripción del diseño del sistema, se detallaron cada una de las tareas que se realizaron en las dos iteraciones de la aplicación. Se definieron los estándares de codificación puestos en práctica en la implementación Además, se pudo constatar que el software no presentaba errores al ser satisfactorias las pruebas unitarias y más importante aún se logró la aceptación del

cliente por los resultados satisfactorios arrojados por cada una de las pruebas de aceptación.

Capítulo IV. Estudio de la factibilidad

Introducción

Este capítulo como parte del proceso de planificación de desarrollo de software tiene una vital importancia, la estimación, la cual consiste en determinar con cierto grado de certeza los recursos necesarios para el desarrollo del mismo, ya sean recursos de hardware, software, esfuerzo, tiempo y costo. Se realizará un estudio de la factibilidad para la realización del software propuesto mediante una estimación de tamaño, esfuerzo y planificación necesaria para llevar a cabo el mismo.

4.1 Modelo Matemático COCOMO II.

COCOMO, propuesto y desarrollado por Barry Boehm, es uno de los modelos de estimación de costos mejor documentados y utilizados. El modelo permite determinar el esfuerzo y tiempo que se requiere en un proyecto de software a partir de una medida del tamaño del mismo expresada en el número de líneas de código que se estimen generar para la creación del producto software.

Consiste básicamente en la aplicación de ecuaciones matemáticas sobre los Puntos de Función sin ajustar o la cantidad de líneas de código (SLOC, Source Lines of Code) estimados para un proyecto. Estas ecuaciones se encuentran ponderadas por ciertos factores de costo (cost drivers) que influyen en el esfuerzo requerido para el desarrollo del software.

Está compuesto por tres modelos que se adaptan tanto a las necesidades de los diferentes sectores descritos, como al tipo y cantidad de información disponible en cada etapa del ciclo de vida de desarrollo, lo que se conoce por granularidad de la información.

El estudio de factibilidad de este sistema se realizó mediante el modelo de diseño temprano que se utiliza en las primeras etapas del desarrollo, en las cuales se evalúan las alternativas de hardware y software de un proyecto. En estas etapas se tiene poca información, se conoce muy poco del tamaño del producto a ser desarrollado, de la naturaleza de la plataforma, del personal a ser incorporado al proyecto o aspectos específicos del proceso a utilizar. Ese nivel de detalle en este modelo es consistente con el nivel general de información disponible y con el nivel general de estimación detallada que es necesaria en estas etapas, lo que concuerda con el uso de Puntos de Función. Para estimar tamaño usa Puntos de Función No Ajustados como métrica de medida y un número reducido de factores de costo.

4.2 Características del sistema.

El primer paso a llevar a cabo para la estimación del proyecto consiste en la obtención de los Puntos de Función desajustados, los cuales están dados por la suma de cada una de las entradas, las salidas y las consultas externas del sistema, así como los archivos lógicos internos y de interfaz externos. A continuación se muestran cada una de estas características aplicadas al software en cuestión.

4.2.1 Entradas externas.

Se definen como un proceso elemental mediante el cual ciertos datos cruzan la frontera del sistema desde afuera hacia adentro. En el caso particular de la aplicación propuesta se cuenta con dos entradas externas, especificadas en la siguiente tabla:

Nombre de la entrada externa.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Cargar imagen CT	1	1	Simple
Cargar imagen SPECT	1	1	Simple
Total	**2**	**2**	

Tabla 28: Entradas externas.

4.2.2 Salidas externas.

Se definen como un proceso elemental con componentes de entrada y de salida mediante el cual datos simples y datos derivados (esto es, datos que se calculan a partir de otros datos) cruzan las fronteras del sistema desde adentro hacia afuera. Las salidas externas vinculadas al proyecto se describen a continuación.

Nombre de la salida externa.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Salvar Imagen CT	1	3	Simple

Salvar Imagen SPECT	1	3	Simple
Visualizar las etiquetas de la imagen CT	1	1	Media
Visualizar las etiquetas de la imagen SPECT	1	1	Media
Salvar las etiquetas de la imagen CT.	1	3	Media
Salvar las etiquetas de la imagen SPECT.	1	3	Media
Crear fichero de entrada al código MCNPX.	2	2	Compleja
Total	**8**	**16**	

Tabla 29: Salidas externas.

4.2.3 Consultas externas.

Se definen como un proceso elemental con componentes de entrada y de salida donde un Actor del sistema rescata datos de uno o más Archivos Lógicos Internos o Archivos de Interfaz Externos. Los datos de entrada no actualizan ni mantienen ningún archivo (lógico interno o de interfaz externo) y los datos de salida no contienen datos derivados (es decir, los datos de salida son básicamente los mismos que se obtienen de los archivos).En este caso no fueron identificados.

Nombre de la Petición.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Total		**0**	

Tabla 30: Consultas externas.

4.2.4 Archivos lógicos internos

Constituyen un grupo de datos relacionados lógicamente e identificables por el usuario, que residen enteramente dentro de los límites del sistema y se mantienen a través de entradas externas.

Nombre de la petición.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Total		**0**	

Tabla 31: Fichero lógico interno.

4.2.5 Archivos de interfaz externos.

Son un grupo de datos relacionados lógicamente e identificables por el usuario, que se utilizan solamente para fines de referencia. Los datos residen enteramente fuera de los límites del sistema y se mantienen por las Entradas Externas de otras aplicaciones, es decir, cada Archivo de Interfaz Externo es un Archivo Lógico Interno de otra aplicación. En este caso fue identificado un Archivos de interfaz externa.

Nombre de la interfaz externa.	Cantidad de ficheros.	Cantidad de elementos de datos.	Clasificación (Simple, Media. Compleja)
Total		**0**	

Tabla 32: Interfaces externas

4.2.6 Puntos de función desajustados

Elementos.	Simple.		Medio.		Complejo.		Subtotal.
	No.	Peso.	No.	Peso.	No.	Peso.	
Entradas externas.	2	3	0	4	0	6	6
Salidas externas.	2	4	2	5	1	7	25

Consultas externas.	0	3	0	4	0	6	0
Fichero lógico interno.	0	7	0	10	0	15	0
Fichero interfaz interno.	0	5	0	7	0	10	0
Total (UFP):							**31**

Tabla 33: Puntos de función desajustados

Se encuentra la tabla de peso del factor de complejidad. [Anexo I]

4.3 Cálculos de instrucciones fuentes, esfuerzo, tiempo de desarrollo, cantidad de hombres y costo.

4.3.1 Cálculo de instrucciones fuentes.

Una vez obtenida la cantidad total de puntos de función desajustados pertenecientes al proyecto, se procede a calcular la cantidad de instrucciones fuentes del mismo, para lo cual se utiliza la ecuación que se muestra seguidamente:

$$\boldsymbol{SLOC = UFP\ x\ ratio}$$

Dónde:

$\boldsymbol{SLOC}$: Cantidad de instrucciones fuente (992).

$\boldsymbol{UFP}$: Puntos de función desajustados. (31)

$\boldsymbol{ratio}$: Conversión de puntos de función desajustados a líneas de código para el lenguaje C#. (32)

Partiendo de la ecuación anterior se obtiene el siguiente resultado:

$$\boldsymbol{SLOC = 31\ x\ 32 = \ 992 = 1\ KSLOC}$$

KSLOC: Tamaño del software a desarrollar.

Las características del sistema y valores obtenidos anteriormente han sido plasmadas en la siguiente tabla:

Características	Valor
Puntos de función desajustados.	31
Lenguaje (C#).	32
Instrucciones fuentes por puntos de función.	1000 SLOC
Instrucciones fuentes.	1 KSLOC

Tabla 34: Características del sistema.

4.3.2 Cálculo de esfuerzo nominal.

Posteriormente se procede al cálculo del esfuerzo nominal, ecuación que se toma como base tanto en el método de diseño preliminar, al cual se hace alusión anteriormente, como en el modelo Post arquitectura, ambos definidos por COCOMO II.

$$\boldsymbol{PM_{Nominal} = Ax(Size)^{B}}$$

$$\boldsymbol{PM_{Nominal} = 2.94\ x\ (1)^{1.06}}$$

$$\boldsymbol{PM_{Nominal} = 2.94\ meses/hombre}$$

Dónde:

$\boldsymbol{PM_{Nominal}}$: Esfuerzo nominal requerido en meses – hombres.

$\boldsymbol{Size}$: Tamaño estimado del software en Puntos de Función sin Ajustar. (1 KSLOC)

$\boldsymbol{A}$: Constante que se utiliza para capturar los efectos multiplicativos en el esfuerzo requerido de acuerdo al crecimiento de tamaño del software. (2.94).

$\boldsymbol{B}$: Constante denominada Factor escalar y su valor está dado por la resultante de los aspectos positivos sobre los negativos que presenta el proyecto. (1.06)

$$\boldsymbol{B = 0.91 + 0.01\ x\ \sum(W_i)}$$

$$\boldsymbol{B = 0.91 + 0.01\ x\ 15.84}$$

$$B = 1.07$$

Dónde:

W_i: Variables escalares que indican las características que el proyecto presenta en lo que a su complejidad y entorno de desarrollo se refiere.

Precedentes (PREC).

El factor de precedencia (PREC) toma en cuenta el grado de experiencia previa en relación al producto a desarrollar, tanto en aspectos organizacionales como en el conocimiento del software y hardware a utilizar.

Flexibilidad de desarrollo (FLEX).

El factor de flexibilidad (FLEX) considera el nivel de exigencia en el cumplimiento de los requerimientos preestablecidos, plazos de tiempos y especificaciones de interfaz.

Cohesión del equipo (TEAM).

El factor de escala denominado Cohesión del Equipo tiene en cuenta las dificultades desincronización entre los participantes del proyecto: usuarios, clientes y desarrolladores. Estas dificultades pueden surgir por diferencias culturales, dificultad en la conciliación de objetivos, falta de experiencia y familiaridad con el trabajo en equipo.

Solución de riesgos (RESL).

Este factor involucra aspectos relacionados al conocimiento de los ítems de riesgo crítico y al modo de abordarlos dentro del proyecto.

Madurez del proceso (PMAT).

El procedimiento para determinar el PMAT es establecer el porcentaje de cumplimiento de cada una de las Áreas evaluando el grado de cumplimiento de las metas correspondientes.

La siguiente tabla muestra los valores asignados a cada una de estas variables:

Nombre.	Valor.	Justificación.
PREC	6.20	Existen varios proyectos similares a nivel internacional, pero no se tiene

		acceso a ellos por cuestiones económicas.
FLEX	1.01	Cuenta con alta flexibilidad en cuanto a los requerimientos establecidos inicialmente.
TEAM	0.0	El equipo de desarrollo presenta una perfecta cohesión.
RESL	7.07	No se identificaron riesgos.
PMAT	1.56	El porciento de cumplimiento respecto a las tareas asignadas es elevado.
Total(SF)	15.84	

Tabla 35: Factores de escala.

Se encuentra la tabla de factores de escala. [Anexo I]

4.3.3 Ajuste del esfuerzo nominal.

El esfuerzo calculado anteriormente es un valor nominal y debe ser ajustado en base a las características del proyecto para lo cual se tiene un conjunto de Multiplicadores de Esfuerzo (ME) que representan las características del proyecto y expresan su impacto en el desarrollo total del producto de software.

4.3.4 Clasificados en categorías.

Los 7 Multiplicadores de Esfuerzo son:

Del Producto.

RCPX: Confiabilidad y Complejidad del producto

RUSE: Reusabilidad Requerida

De la Plataforma.

PDIF: Dificultad de la Plataforma

Del Personal.

PERS: Aptitud del Personal

PREX: Experiencia del Personal

Del Proyecto.

FCIL: Facilidades

SCED: Cronograma de Desarrollo Requerido

Nombre.	Valor.	Justificación.
RCPX	2.38	La complejidad del sistema es muy alta.
RUSE	1.00	Pretende reutilizarse una parte del código.
PDIF	1.00	Uso de memoria y almacenamiento normal, plataforma estable.
PREX	1.12	Baja experiencia en el personal en cuanto a la utilización del lenguaje y herramientas.
PERS	1.00	La capacidad del personal es normal.
FCIL	1.00	La utilización de entornos de desarrollo integrados facilita en gran medida el trabajo.
SCED	1.00	El sistema se desarrolló en el tiempo establecido.
Total(EM)	8.5	

Tabla 36: Multiplicadores de esfuerzo.

Se encuentra la tabla de multiplicadores de esfuerzo del modelo del diseño temprano. [Anexo I].

$$PM_{ajustado} = PM_{Nominal}\ x \prod (ME_t)$$

$$PM_{ajustado} = 2.94 \frac{meses}{hombres} x\ 8.5$$

$$PM_{ajustado} = 25\ meses/hombres$$

4.3.5 Cálculo del tiempo de desarrollo del software.

El tiempo requerido para el desarrollo del proyecto está dado por la siguiente ecuación:

$$TDEV = 3.64\ x\ (PM_{ajustado})^F$$

$$TDEV = 3.64\ x\ (25)^{0.27}$$

$$TDEV = 8.68$$

Dónde:

C: Constante con valor 3.64.

$$PM_{ajustado} = 25\ meses/hombre$$

F: 0.27.

CALCULAR

$$F = D + 0.2\ x\ 0.01\ x \sum SF$$

$$F = 0.24 + 0.2\ x\ 0.01\ x\ 15.84$$

$$F = 0.27$$

Dónde:

D: Constante cuyo valor es 0.24.

SF: Valor de los factores de escala (15.84).

4.4 Cálculo del costo total del proyecto.

Para el cálculo del costo total correspondiente al proyecto en cuestión, COCOMO II propone la siguiente ecuación:

$$\boldsymbol{C = CHM\ x\ PM}$$

$$\boldsymbol{C = 1440\ x\ 25}$$

$$\boldsymbol{C = 36000}$$

CALCULAR

Dónde:

C: Costo total.

$\boldsymbol{CHM}$: Costo teniendo en cuenta salario de todos los obreros, el cual se calcula por la siguiente ecuación.

$$\boldsymbol{CHM = CH\ x\ sal}$$

$$\boldsymbol{CHM = 2.88\ x\ 500}$$

$$\boldsymbol{CHM = 1440}$$

CALCULAR

Dónde:

$\boldsymbol{sal}$: Salario medio por cada trabajador (500).

CH: Cantidad de personas destinadas al proyecto, lo que es calculado a través de la ecuación:

$$\boldsymbol{CH = PM/TDEV}$$

$$\boldsymbol{CH = 25/8.68}$$

$$\boldsymbol{CH = 2.88}$$

4.5 Resultados.

En la siguiente tabla se muestran los resultados obtenidos luego de haber efectuado todos los cálculos para determinar el costo y esfuerzo requeridos por el proyecto.

Cálculo de:	Valor

Esfuerzo.	3 meses/hombre.
Tiempo de desarrollo.	9 meses.
Cantidad de hombres.	3 hombres.
Salario medio.	500 pesos.
Costo.	36000 pesos.

Tabla 37: Resultados.

4.6 Análisis de costo.

El desarrollo de un producto siempre tiene un costo de producción, el cual debe ser justificado en base a los beneficios reportados por el mismo. El sistema que se propone en este trabajo no conlleva a grandes gastos, puesto que solo es influyente el salario de los desarrolladores, por lo cual se concluye que su implementación es factible.

4.7 Conclusiones.

En el presente capítulo se realizó un análisis de factibilidad de la solución propuesta, arribando a la conclusión de que es viable su desarrollo comparando los costos de producción con los beneficios reportados por su puesta en funcionamiento.

Conclusiones Generales

Con el desarrollo de este trabajo se profundizó en el conocimiento del formato DICOM y la Imagenología. Se aplicó la metodología XP para guiar el proceso de desarrollo de software, mediante el cual se obtuvo la implementación de un programa de interfaz para el código MCNPX, con las funcionalidades previstas para el primer ciclo de desarrollo del software. El diseño y la implementación se rigieron por algunas herramientas como Visual Studio 2010, .NET Framework para la programación. Se puede concluir que se ha cumplido satisfactoriamente el objetivo trazado para este trabajo, enfatizando en los siguientes puntos:

- ✓ Se implementó una interfaz que provee un fichero de entrada para el MCNPX.
- ✓ Se desarrolló una interfaz amigable que permite analizar toda la información de las imágenes DICOM automatizando esta tarea para el personal de salud pública.

Recomendaciones.

A continuación son listadas las recomendaciones que en opinión del autor deben ser tenidas en cuenta con el objetivo de realizar un seguimiento de este trabajo:

1. Permitir el análisis de otros formatos de imágenes como Interfile y Analize.

2. Migrar la aplicación a otros sistemas operativos, ya que con el advenimiento del desarrollo informático nuestro país se encuentra en una fase de migración hacia el software libre.

3. Reimplementar el software realizado para utilizar procesamiento con tarjetas de video o unidades de procesamiento gráfico, ya que son más potentes para el trabajo con imágenes.

Referencias Bibliográficas

1. Guy, M.J., et. Al., RMDP-MC: A dedicated package for I-131 SPECT quantification, patient-specific dosimetry and Monte Carlo, Proc. Int. Symp. Nashville, TN, (2002).

2. Kolbert KS, Sgouros G, Scott AM, et al. Implementation and evaluation of patient-specific three-dimensional internal dosimetry. J Nucl Med. 1997;38: 301-308.

3. Liu A, Williams LE, Lopatin G et al. A radionuclide therapy treatment planning and dose estimation system, J. Nucl. Med., 40, 1151–3, 1999.

4. Clairand I., M. Ricard, J.Gouriou, M. Di.Paola and B. Aubert, DOSE3D: EGS4 Monte Carlo code-based software for internal radionuclide dosimetry, J.of Nuclear Medicine, Vol 40, Issue 9, 1517-1523, (1999).

5. Yoriyaz, H.; Stabin, M.G.; Santos, A. Monte Carlo MCNP-4B-Based Absorbed Dose Distribution Estimates for Patient-Specifc Dosimetry. J Nucl Med, 42:662-669, 2001.

6. Tagesson M, Ljungberg M, Strand S-E. The SIMDOS Monte Carlo code for conversion of activity distributions to absorbed dose and dose-rate distributions. In: Stelson A, Stabin M, Sparks R, eds. Sixth International Radiopharmaceutical Dosimetry Symposium Oak Ridge, TN: Oak Ridge Associated Universities; 1999:425–440.

7. Milián Felix M y Gual Maritza R., IMAGON3D, Programa registrado en CENDA, C. Habana, Cuba, No. de registro: 2560-2004, (2004).

8. Brown, 2003, MCNP5 - Monte Carlo N Particle, Origin : USA, 1940/50. Los Alamos Nat. Lab.

9. Oualline, S. Practical C++ Programming. O'Reilly and Associates, Inc. Pag: 3-7. 1995.

10. Charte,F. Visual C#.Net. Anaya Multimedia, SA. 2002.

11. Smith, E. A., Whisler, V., et al. Visual Basic 6 Bible. IDG Books WorldWide, Inc. USA, 1998.

12. Deitel, H. M. y Deitel, P. J. Java How to Program. Prentice Hall. 2002.

13. Robinson, S., Cornes, O., et al. Professional C#. Wrox Press Ltd. 2001.

14. http://URRIELLU.net

15. Garcia, J. (n.d.). Patrones de diseño. Retrieved 5 4, 2011, from Ingeniero de software:http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

Bibliografía

1. L. Bouwens, M. Koole y Y. D'Asseler. *Image-correction techniques in SPECT*, 2001.
2. Stolberg,. HO., Nahmias,. C. Challenges and opportunities in diagnostic imaging: A Canadian perspective. Canadian Association of Radiologists Journal, 2002. 53 (3): 130-132.
3. Birkfellner W., Applied Medical Image Processing, A Basic Course, Taylor & Francis Group, 2011.
4. Waters, L.S.(ed.), "MCNPX, version 2.4.0", LA CP-02-408 (Los Alamos National Laboratory, Los Alamos, NM, September, 2002.
5. Kolbert K S et al. J. Nucl. Med., 38, 301–8, 1997.
6. Liu A et al. J. Nucl. Med., 40, 1151–3, 1999.
7. Clairand I et al. J. Nucl. Med. 40, 1517–23, 1999.
8. Dewaraja Y K et al. J. Nucl. Med., 46, 840–9, 2005.
9. Ljungberg, M. and Strand, S.-E., A Monte Carlo Program Simulating Scintillation Camera Imaging, Comp. Meth. Progr. Biomed. 29, 257-272, 1989
10. S. JAN et. al. "GATE: a simulation toolkit for PET and SPECT", Phys. Med. Biol. 49, 4543, 2004.
11. http://www.monografias.com/trabajos24/herramientas-case/herramientas-case.shtml
12. Kolbert K S et al. J. Nucl. Med., 38, 301–8, 1997.
13. Liu A et al. J. Nucl. Med., 40, 1151–3, 1999.
14. Clairand I., M. Ricard, J.Gouriou, M. Di.Paola and B. Aubert, DOSE3D: EGS4 Monte Carlo code-based software for internal radionuclide dosimetry, J.of Nuclear Medicine, Vol 40, Issue 9, 1517-1523, (1999).
15. Guy, M.J., et. Al., RMDP-MC: A dedicated package for I-131 SPECT quantification, patient-specific dosimetry and Monte Carlo, Proc. Int. Symp. Nashville, TN, (2002).
16. Dewaraja Y K et al. J. Nucl. Med., 46, 840–9, 2005.
17. Ljungberg, M. and Strand, S.-E., A Monte Carlo Program Simulating Scintillation Camera Imaging, Comp. Meth. Progr. Biomed. 29, 257-272, 1989

18. S. JAN et. al. “GATE: a simulation toolkit for PET and SPECT”, *Phys. Med. Biol.* 49, 4543, 2004. Petoussi-Henss, N. *et al* The GSF family of voxel phantoms, *Phys. Med. Biol.* 47, 89, 2002
19. Zubal IG, Harrell CR, Smith EO, Rattner Z, Gindi G and Hoffer PB, Computerized 3-Dimensional Segmented Human Anatomy Med. Phys. 21(2): 299-302, 1994.
20. Xu, X. G., Chao, T. C. and Bozkurt, A. VIP-Man. An image-based whole-body adult male model constructed from color photographs of the Visible Human Project for multi-particle Monte Carlo calculations. Health Phys. 78, 476–486 (2000).
21. M. Cristy and K. F. Eckerman. Specific Absorbed Fractions of Energy At Various Ages from Internal Photon Sources. II. One-Year-Old, 1987, ORNL/TM-8381/V2**.**
22. K. F. Eckerman, M. Cristy, and J. C. Ryman, “The ORNL Mathematical Phantom Series,” Oak Ridge National Laboratory Report, available at http://homer.hsr.ornl.gov/VLab/VLabPhan.html (Dec. 1996).
23. W. S. Snyder, M. R. Ford, G. G. Warner, and S. B. Watson, "A Tabulation of Dose Equivalent per Microcurie-day for Source and Target Organs of an Adult for Various Radionuclides: Part I," Oak Ridge National Laboratory Report ORNL-5000 (1974).
24. Yoriyaz Helio, dos Santos Admir and Stabin Michael G., Monte Carlo MCNPX Based Absorbed Dose Distribution Estimates for Patient-Specific Dosimetry, The Journal of Nuclear Medicine, Vol. 42 No. 4, pp. 662-669, April (2001).
25. Milian M Felix, Garcia Fermin, Yoriyaz H, Siqueira P.T. D. and Sena I., TOMO_MC programa para criação de input files para o MCNPX a partir de modelos anatomicos 3D, XIII Seminário de Inicaição Cientifica e 9ª Semana de Pesquisa e Pós-Gradiuação da UESC Ciencias Exatas, da Terra e Engenherias, (2008).
26. Zaidi, H. and Xu, X. G. Computational anthropomorphic models of the human anatomy: the path to realistic Monte Carlo modeling in radiological sciences, Ann. Rev. Biomed. Eng. 9, 1.1–1.30 (2007).
27. Bas Revert, “DICOM Cook book” Phillips Medical Systems (1997).
28. Milián Felix M y Gual Maritza R., IMAGON3D, Programa registrado en CENDA, C. Habana, Cuba, No. de registro: 2560-2004, (2004).
29. International Commission on Radiation Units and Measurements. Conversion photon, electron, proton and neutron interaction data for body tissue. ICRU Report 46 (Bethesda, MD, USA) (1992).
30. Kramer R, Khoury H J, Vieira JWand Lima V J M, MAX06 and FAX06: update of two adult human phantoms for radiation protection dosimetry Phys. Med. Biol. 51 3331–46, 2006

Glosario de términos

DICOM: Digital Imaging and Communications in Medicine

SPECT: Single Photon Emission Computed Tomography.

CT: Computed Tomography .

PET: Positron Emission Tomography.

MCNPX: Monte Carlo N-Particle X.

MIRD: Medical Internal Radiation Dosimetry.

GRASP: En diseño orientado a objetos, GRASP son patrones generales de software para asignación de responsabilidades, es el acrónimo de "General Responsibility Assignment Software Patterns".

XP: Extreme Programming.

WPF: Windows Presentation Fundation.

LINQ: Language Integrated Query.

3D: Tridimensional, compuesto por tres dimensiones.

Voxel: Unidad de medida de Imágenes y objetos tridimensionales en la informática.

Pixel: Unidad de medida de elementos visuales en la informática, muy aplicable a las imágenes.

UML: Lenguaje de Modelado Unificado.

HU: Historias de Usuario.

RUP: Rational Unified Process.

Cálculo Dosimétrico: Determinación de la dosis absorbida en los tejidos sanos y tumorales.

Dosis absorbida: Energía depositada por unidad de masa

Medicina Nuclear: Es una técnica por la cual se obtienen imágenes funcionales inyectando un radionúclidos (isótopo de origen natural o artificial que muestra radiactividad) al paciente, cuyas emisiones son recogidas en una cámara para fines de diagnóstico y tratamiento.

Radionúclido: Compuesto radioactivo (emite radiaciones)

Radiofármaco: Fármaco que contiene en su molécula un átomo radioactivo y que se utiliza con fines diagnósticos, de investigación o terapéuticos

Imagen médica: Es la representación de la distribución espacial de una o más propiedades físicas o químicas dentro del cuerpo humano.

Modalidades de imagen médica: Son las diferentes técnicas utilizadas para su obtención.

Corregistradas: Un punto determinado de la imagen A corresponde al mismo punto en la imagen B. Pueden ser imágenes de diferentes modalidades, pero necesariamente han de ser del mismo paciente.

Monte Carlo: Es un método estadístico que permite resolver problemas físicos y matemáticos mediante la simulación de variables aleatoria

Anexos I. Tablas de Cocomo II.

Tipo de función	Peso del Factor de Complejidad		
	Bajo	Promedio	Alto
Entradas Externas (Inputs)	3	4	6
Salidas Externas (Outputs)	4	5	7
Archivo Lógicos Internos (Archivos)	7	10	15
Archivos Externos de Interfase (Interfases)	5	7	10
Consultas Externas (Queries)	3	4	6

Ilustración 5: Peso del factor de complejidad

Factores de Escala (Wi)	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.84	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

Tabla 38: Factores de escala de Cocomo II

	Extra bajo	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extra alto
RCPX	0.73	0.81	0.98	1.00	1.30	1.74	2.38
RUSE	--	--	0.95	1.00	1.07	1.15	1.24
PDIF	--	--	0.87	1.00	1.29	1.81	2.61
PERS	2.12	1.62	1.26	1.00	0.83	0.62	0.50
PREX	1.59	1.33	1.12	1.00	0.87	0.71	0.62
FCIL	1.43	1.30	1.10	1.00	0.87	0.73	0.62
SCED	--	1.43	1.14	1.00	1.00	1.00	--

Tabla 39: Multiplicadores de esfuerzo Cocomo II