Universidad de las Ciencias Informáticas

Facultad 5 "Entornos Virtuales"

Propuesta del proceso de desarrollo del software del Simulador Quirúrgico.

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

Autor(as): Daylín Santos Urquiaga.
Liena La Rosa Castro.

Tutor(as): Ing. Mariela Nogueira Collazo.
Ing. Yuremis Mengana Claro.

Asesora: Ing. Yusleidy Guelmes León.

Ciudad de La Habana, julio 2008.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____	_____
Daylín Santos Urquiaga	Liena La Rosa Castro
(Autora)	(Autora)
_____	_____
Ing. Yuremis Mengana Claro	Ing. Mariela Nogueira Collazo
(Tutora)	(Tutora)

DATOS DE CONTACTO

Nombre y apellidos: Mariela Nogueira Collazo
Institución: UCI
Título: Ingeniera en Ciencias Informáticas
Categoría Docente: Adiestrado
Año de Graduación: 2007
E-mail: mnogueira@uci.cu
Miembro del proyecto productivo SMQ.

Nombre y apellidos: Yuremis Mengana Claro
Institución: UCI
Título: Ingeniera en Ciencias Informáticas
Categoría Docente: Adiestrado
Año de Graduación: 2007
E-mail: ymengana@uci.cu
Miembro del proyecto productivo SCADA.

Nombre y apellidos: Yusleidy Guelmes León.
Institución: UCI
Título: Ingeniera Informática
Categoría Docente: Instructor
Año de Graduación: 2005
E-mail: yguelmes@uci.cu
Actividades y cargos desempeñados: Ha impartido clases de Gestión de Software, Bases de Datos e Ingeniaría de Software. Se ha desempeñado como asesora de Arquitectura de la facultad 5. Se encuentra cursando la maestría de Gestión de Proyectos Informáticos.

Para lograr el triunfo siempre ha sido indispensable
pasar por la senda de los sacrificios.

Simón Bolívar

A mi abuela Ana porque supo esperar pacientemente y con fe este triunfo.
A mis padres por darme la vida y más.

"Liena"

A mis padres que son mi razón de ser.
A mis hermanos por traer tanta felicidad a mi vida.

"Daylin"

Es muy importante para mí agradecer desde lo más profundo de mi corazón a todas las personas que estuvieron a mi lado y confiaron en mí a lo largo de todo este difícil camino. Es ineludible que sepan que significan para mí, muchísimo más de lo que pueda quedar reflejado en un papel…

- ✓ *A mis abuelos por confiar en mí.*
- ✓ *A mis padres, por todo su apoyo incondicional, por sus desvelos, en fin, por todo su "amor" que ha sido tan importante en mi vida y en mi formación profesional, para ellos todo mi amor y agradecimiento.*
- ✓ *A mi hermanita bella, que ha sido y seguirá siendo una guía insustituible en mi vida.*
- ✓ *A mis "hermanas por elección" Lisa y Daylín que la vida las puso en mi camino para darme muchas más alegrías y triunfos, porque he aprendido mucho de ellas.*
- ✓ *A mi tía preferida por ser como mi madre y quererme y apoyarme como tal.*
- ✓ *A mis tíos y primos por su cariño infinito.*
- ✓ *A mis bebés Cynthia, Hache y Avi por sus calurosos recibimientos y las alegrías que me dan.*
- ✓ *A Mariela por entenderme y soportarme, por enseñarme a trabajar organizado, a ser optimista, en fin por enseñarme a hacer un proceso de desarrollo…*
- ✓ *A mi compañera de tesis que sin su entrega constante y su paciencia para conmigo no hubiera podido realizar este trabajo.*
- ✓ *A la familia de Daylín por quererme y enseñarme que la familia va más allá de los lazos sanguíneos.*
- ✓ *A mi queridísimo grupo, a las nenas por aguantarme 5 años con este carácter, a los nenes por cargar conmigo siempre y a los que ya no están…, a todos los adoro y siempre los recordaré.*
- ✓ *A mis amistades y profesores por educarme, por los buenos y malos momentos por los que aún me ayudan a transitar en la vida.*

A todos GRACIAS por existir.

"Liena"

- ✓ *A mi tutora Mariela, por su ayuda incondicional, su paciencia y las noches de desvelo. Muchas Gracias.*
- ✓ *A mis padres, que en los momentos difíciles supieron confiar en mí y lo han dado todo para que sea una mujer de bien, por su apoyo incondicional y su ejemplo. Por guiarme en cada paso de mi vida, por ser los mejores padres del universo. A ustedes les estaré eternamente agradecida.*
- ✓ *A mis hermanos, por ser las personas más especiales en mi vida, por quererme tanto, por preocuparse por mí.*
- ✓ *A mi novio Leandro, por llegar a mi vida y brindarme todo su apoyo. Por enseñarme a confiar en mí. Por ser una persona tan especial.*
- ✓ *A mi tía Ada, por su cariño y preocupación, por ser mi segunda mamá.*
- ✓ *A mi abuelo Cheo, por estar todo el tiempo a mi lado. Te quiero mucho.*
- ✓ *A mi abuela Julia, que aunque no estuvo presente es mis luchas de estudio, siempre estará en mi corazón.*
- ✓ *A toda la familia, por su cariño y preocupación.*
- ✓ *A Liena, amiga y compañera de tesis, que supo estar conmigo en los momentos buenos y malos. Por soportarme en esta contienda de sacrificio. Gracias por permitirme ser tu amiga.*
- ✓ *A la familia de Liena por su cariño en estos años de carrera, por acogerme como una más.*
- ✓ *A las amistades en estos años de estudio, por los momentos vividos.*
- ✓ *A mis compañeros de grupo con los cuales he pasado cinco años de sacrificio y alegrías.*
- ✓ *A Parra y el Flaco "mis pequeños profes", que me enseñaron a programar.*

"Daylín"

RESUMEN

La industria del software se ha desarrollado notablemente en los últimos años, debido a esto, cada producto requiere de un proceso específico que esté fundamentado en las características del proyecto, el entorno de desarrollo y las necesidades del cliente, que garantizan la efectividad de las actividades de desarrollo y la calidad de la solución obtenida.

Esta investigación surge por la necesidad de organizar el proceso productivo del proyecto “Simulador Quirúrgico” (SMQ) en la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI), el cual tiene la misión de elaborar un Simulador para el entrenamiento de la Cirugía de Mínimo Acceso (CMA) utilizando la Realidad Virtual (RV). De aquí que, el objetivo de este trabajo es: definir el proceso de desarrollo de software para el Simulador Quirúrgico desarrollado en el proyecto SMQ. Para lograr esto, se analizaron diferentes metodologías de desarrollo de software, tomando de ellas características que se pueden implementar en el proceso propuesto y que contribuyen a su perfeccionamiento.

Se realiza un análisis de las etapas y actividades que estaban definidas hasta el momento en el proyecto, lo cual sirve de base para diseñar el nuevo proceso. Se define en este trabajo una primera versión de la plantilla que servirá como guía para elaborar el guión técnico del Simulador Quirúrgico.

El proceso definido está dividido en tres fases, una primera de Conceptualización y Planificación, que tiene como objetivo definir los objetivos y el alcance del proyecto y a partir de esto se planifican los recursos y se establece el cronograma de producción; una segunda, que enmarca las actividades concretas para desarrollar el software y la última, se centra en las actividades necesarias para garantizar la entrega final del producto al cliente.

Palabras Claves

Proceso de desarrollo, Metodología de desarrollo, Simulador Quirúrgico, Realidad Virtual

INTRODUCCIÓN

El avance de la informática ha propiciado en los últimos años el surgimiento de un nuevo término: “Realidad Virtual (RV)”. Este concepto encierra amplias aplicaciones que van encaminadas a simular objetos y procesos del mundo real. Los Sistemas de Realidad Virtual (SRV), se han expandido considerablemente a diferentes sectores de la sociedad en los últimos años. Esta tecnología puede encontrarse en aplicaciones de la defensa, la medicina, la educación y el entretenimiento.

Una de las aplicaciones más fomentadas de los SRV son los simuladores. La simulación se ha potenciado en varios campos, ya existen diferentes productos que reflejan las potencialidades de esta técnica, dentro de los más difundidos están: los simuladores de conducción de autos, de trenes, de barcos, de aviones, de tiro, también hay otros más enfocados al entretenimiento como es el simulador de montaña rusa, y otros que requieren niveles muy elevados de realismo como es el caso de los simuladores de Cirugía de Mínimo Acceso (CMA).

La CMA, ha sustituido a la cirugía abierta en un número considerable de procederes. Los procedimientos de cirugía general, *colecistectomía*, cirugía anti reflujo, cirugía del colon, cirugía de rodilla y hombro, son desarrollados a partir de estas técnicas y se está produciendo un incremento de estas aplicaciones en los campos de ginecología, cirugía cardíaca y urología.

Los Simuladores Quirúrgicos se emplean para facilitar el aprendizaje de los cirujanos que se adiestran en esta especialidad. En la mayoría de los centros de entrenamiento los especialistas se auxilian de cuerpos inanimados como por ejemplo: maniquíes y cadáveres para probar y desarrollar sus habilidades, esto constituye un problema dado que los órganos y los tejidos de estos no presentan un comportamiento real, también se utilizan animales vivos o intervenciones quirúrgicas reales supervisadas por especialistas, lo cual tampoco resulta una buena alternativa, pues no permite deshacer o repetir acciones hasta adquirir la habilidad necesaria. La simulación quirúrgica permite además, que el cirujano pueda adiestrarse en patologías poco frecuentes o severas.

En la Universidad de las Ciencias Informáticas (UCI) se lleva a cabo el Proyecto Productivo SMQ, en convenio con el Ministerio de Salud Pública y otras empresas nacionales, que persiguen el objetivo de desarrollar un simulador para el entrenamiento de la CMA utilizando la RV. Dicho proyecto se encuentra inmerso en la primera etapa de desarrollo y aún sus desarrolladores no cuentan con un

proceso de desarrollo del software bien definido, que se adecue a las características y necesidades del producto y guíe todas las etapas de desarrollo.

A partir de los antecedentes citados surge el **Problema científico**:¿Cómo organizar y definir las actividades y etapas de desarrollo de un Simulador de Cirugía de Mínimo Acceso (CMA) en el marco del Proyecto Productivo SMQ de la Facultad 5?

Una vez planteado este problema científico se determina como **Objeto de estudio** de la investigación: los procesos y las metodologías de desarrollo del software. Y se define como **Campo de acción**: el proceso de desarrollo del software aplicable al Simulador Quirúrgico del Proyecto SMQ.

Para esto se plantea el siguiente **Objetivo general**: definir el proceso de desarrollo del software para un Simulador Quirúrgico, que será empleado en el Proyecto SMQ.

Para desarrollar esta investigación se establecen las siguientes **Tareas investigativas:**

- Realizar un estudio del estado del arte de los Simuladores Quirúrgicos en el mundo.
- Analizar estándares y modelos que se empleen para definir procesos de desarrollo del software.
- Analizar los rasgos distintivos de algunas metodologías de desarrollo del software más utilizadas en los Sistemas de Realidad Virtual.
- Identificar cada una de las etapas de desarrollo del software para el SMQ.
- Determinar las entradas y salidas de cada etapa.
- Definir el flujo de actividades de cada etapa.
- Determinar los roles participantes en cada etapa de desarrollo.
- Identificar los elementos de las metodologías de desarrollo del software que pueden ser acoplados al proceso que se definirá y que contribuyan a su mejoramiento.

El trabajo de diploma está comprendido por 3 capítulos, que estructuran el contenido de la siguiente forma:

En el Capítulo 1 “Fundamentos teóricos a tener en cuenta para definir el proceso de desarrollo de un Simulador Quirúrgico”, se realiza un análisis bibliográfico para adentrarse en el mundo de los

Simuladores Quirúrgicos, la situación de estos en Cuba y finalmente se estudian los modelos de procesos del software y la Norma ISO/IEC 12207.

En el Capítulo 2 “Metodologías de desarrollo”, se realiza un estudio de las características generales de algunas de las metodologías tradicionales y ágiles siguiendo el criterio de expertos, donde se exponen sus características.

En el Capítulo 3 “Propuesta del proceso de desarrollo”, se describen los elementos de las metodologías que se pueden integrar al proceso propuesto y se desglosan las fases del proceso de desarrollo del SMQ que se propone.

Finalmente, se ofrece un glosario de abreviaturas y uno de términos que facilitan la comprensión del lenguaje técnico utilizado en el trabajo.

CAPÍTULO 1. FUNDAMENTOS TEÓRICOS A TENER EN CUENTA PARA DEFINIR EL PROCESO DE DESARROLLO DE UN SIMULADOR QUIRÚRGICO.

En este Capítulo se abordarán algunos conceptos para facilitar la comprensión del tema a desarrollar en la tesis en cuestión. También se detallarán las características de los Simuladores Quirúrgicos y los principales avances en el desarrollo del hardware y/o software de RV aplicados a la laparoscopía. Se realizará además, un estudio de los elementos fundamentales que intervienen en la definición de un proceso de desarrollo del software. Todo lo analizado en este Capítulo conformará la base cognoscitiva para desarrollar la solución deseada.

1.1 Conceptos fundamentales.

¿Qué es Realidad Virtual?

Aunque no existe una definición totalmente aceptada de RV se puede decir que consiste en una simulación tridimensional interactiva por computador en la que el usuario se siente introducido en un ambiente artificial, y que lo percibe como real basado en estímulos a los órganos sensoriales (Parra Márquez, y otros, 2001).

¿Qué es la simulación?

Se entiende por simulación al proceso de diseñar un modelo de un sistema real y llevar a término experiencias con el mismo, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias dentro de los límites impuestos por un cierto criterio o un conjunto de ellos para el funcionamiento de los sistemas (08Fe1).

¿Qué es un Simulador Quirúrgico?

Un Simulador Quirúrgico es la herramienta de software que visualiza las imágenes de entrenamiento contenidas en un determinado proyecto y que permite al cirujano interactuar con el entorno a través de los dispositivos *hápticos* que proporcionan un *feedback* muy realista. El simulador permite todos los

movimientos realizados en laparoscopía: cortar con tijeras, rasgar con garfio, división de tejidos, agarre, colocar grapas, etc. (08En1).

¿Qué es la cirugía laparoscópica?

La cirugía laparoscópica es una técnica quirúrgica que se practica a través de pequeñas incisiones, usando la asistencia de una cámara de video que permite al equipo médico ver el campo quirúrgico dentro del paciente y accionar en el mismo (2008).

¿Qué es la cirugía endoscópica?

La cirugía endoscópica se realiza con pequeñas incisiones, a través de las cuales se introducen sistemas ópticos de televisión e instrumental quirúrgico especialmente diseñado. Puede aplicarse en la cavidad abdominal (*laparoscopía*), en la cavidad torácica (*toracoscopía*, *mediastinoscopía*) o en las partes blandas (cirugía de columna, cirugía plástica, cirugía cervical, etc.) (08Fe).

1.2 La RV en el entrenamiento quirúrgico.

La cirugía es tan antigua como la civilización así como los métodos de aprendizaje empleados. Durante el último milenio, las técnicas y prácticas quirúrgicas han evolucionado considerablemente, aunque aproximadamente desde el año 1.550 a.C. hasta la actualidad, el conocimiento de las técnicas quirúrgicas se trasmite mediante el entrenamiento a aprendices durante intervenciones quirúrgicas reales que son supervisadas continuamente por un experto.

Por esa razón uno de los retos principales de las nuevas tecnologías aplicadas a la medicina es la creación de entornos virtuales para el entrenamiento a cirujanos. Desde hace algunos años y hasta la actualidad se han logrado y registrado avances muy importantes y definitorios en la simulación de CMA para escenarios tales como la cirugía *laparoscópica*, la endoscópica y la artroscópica, entre otras.

Los Simuladores Quirúrgicos constan de modelos virtuales de distintas partes del cuerpo y herramientas virtuales con las que los cirujanos pueden entrenarse en distintas técnicas quirúrgicas. Algunos utilizan texturas fotográficas e imágenes de vídeo de operaciones reales, para lograr una adecuada ambientación del escenario. La posición y la orientación de los instrumentos virtuales

simulan perfectamente la de instrumentos quirúrgicos reales. Mediante ellos los cirujanos se pueden entrenar en técnicas como la cateterización ventricular, inserción de tornillos, punciones lumbares, biopsias de hígado, colocación de trócares y canulación venosa central, entre otras (08Fe1).

Los simuladores médicos en general y en particular los quirúrgicos, son empleados como se dijo anteriormente en este Capítulo, para entrenar a los cirujanos en determinadas técnicas, garantizando así que el aprendiz adquiera la destreza necesaria para enfrentar una situación similar con un paciente real, por tanto, de la calidad de estos sistemas depende que los cirujanos se capaciten correctamente y que logren cumplir con el deber social que les infiere su profesión: salvar vidas humanas.

1.2.1 Ventajas y desventajas que provoca el empleo de los Simuladores Quirúrgicos en el proceso de entrenamiento de la cirugía.

Los simuladores para cirugía laparoscópica como herramientas de aprendizaje proporcionan a los cirujanos las siguientes ventajas:

- La adquisición de habilidades específicas.
- La posibilidad de prácticas repetidas y de bajo costo para evaluar el desempeño.
- Entrenamiento sin riesgos y en un ambiente relajado, practicando a solas o con el operador del laparoscopio habitual, sin tener que estar en un lugar especialmente diseñado para ello.
- Facilidad de acceso a ciertos órganos.
- Ausencia de trauma a la pared abdominal.
- Reducir los costes necesarios con la utilización de cadáveres y animales vivos en el entrenamiento para cirugía.
- Permite la posibilidad de repetir los procedimientos quirúrgicos tantas veces como sea necesario hasta el correcto aprendizaje.
- Permite revisualizar los procedimientos realizados con el objetivo de estudiar sus ventajas e incluso mejorarlo mediante la utilización de técnicas diferentes a las empleadas.
- Permite la planificación y práctica sobre la anatomía de un paciente específico previo a su intervención quirúrgica real.
- Permite el entrenamiento en patologías poco frecuentes.

Desventajas que presenta este tipo de simulador:

- Requieren grandes inversiones en instrumental.
- Período de entrenamiento más largo.
- Requiere de potentes modelos físicos-matemáticos que permitan que los órganos presentes en el entorno de simulación se comporten tal y como lo harían en la realidad al ser sometidos a fuerzas o al ser cortados o cosidos.
- Requiere de potentes modelos geométricos realistas que garanticen la visualización de órganos de la forma más realista posible.

Inconvenientes para el cirujano a la hora de entrenar con un simulador:

- El cirujano cuenta con una visión restringida del campo operatorio.
- La manipulación de los instrumentos es difícil cuando está aprendiendo.
- Presenta una movilidad muy restringida.
- Requiere de buena coordinación mano-ojo.

1.3 Requerimientos de los Simuladores Quirúrgicos.

Para validar el funcionamiento correcto de un simulador se han establecido requerimientos con los que deben cumplir todos los sistemas de este tipo, estos son:

- Lograr una visualización realista de los objetos en la escena (órganos y otras partes internas del paciente).
- Simular de forma realista y en tiempo real, las interacciones de los usuarios con el entorno virtual, respetando las restricciones existentes en la realidad.
- Lograr que los objetos del entorno respondan, mediante modificaciones estructurales realistas a las acciones típicas quirúrgicas como cauterización, corte, sutura, grapado, entre otras, respetando las propiedades físicas de los órganos reales.

Las técnicas de RV tienen por objetivo crear de forma artificial un entorno en el que el usuario pueda ver y sentir los objetos tal y como los vería y sentiría en la realidad. El grado de inmersión en un entorno virtual depende totalmente de la fidelidad con la que este reproduce la realidad que el usuario conoce.

Identificar los siguientes elementos constituye un paso básico para el desarrollo de un simulador que emplee la RV, en estos aspectos se integran las funcionalidades del sistema y engloban los requisitos del mismo:

- Las interfases de comunicación entre el usuario y su entorno virtual.
  - Qué sentidos están implicados en la percepción del entorno virtual.
  - Qué sensores se necesitan para interactuar con el entorno y para proveer la realimentación de fuerzas al usuario.
- Los componentes del entorno 3D y sus propiedades.
- Las interacciones que se podrán establecer entre el usuario y el entorno y cómo este último se debe comportar en base a dichas interacciones.

1.4 Implementación de los Simuladores Quirúrgicos.

La simulación de cirugía, como cualquier otro tipo de simulación, consiste en la ejecución repetitiva de un determinado algoritmo. Como se muestra en la Fig. 1 , durante cada uno de estos ciclos se ha de leer la posición de los instrumentos de interacción, detectar las posibles interacciones de cada uno de estos instrumentos respecto a otros y de los objetos presentes en el entorno, calcular la respuesta física de los tejidos a dichas interacciones y la fuerza de realimentación que esta deformación genera, dibujar la escena resultante y realimentar las fuerzas que ejercen los dispositivos de interacción en función de las fuerzas de realimentación obtenidas(2002).

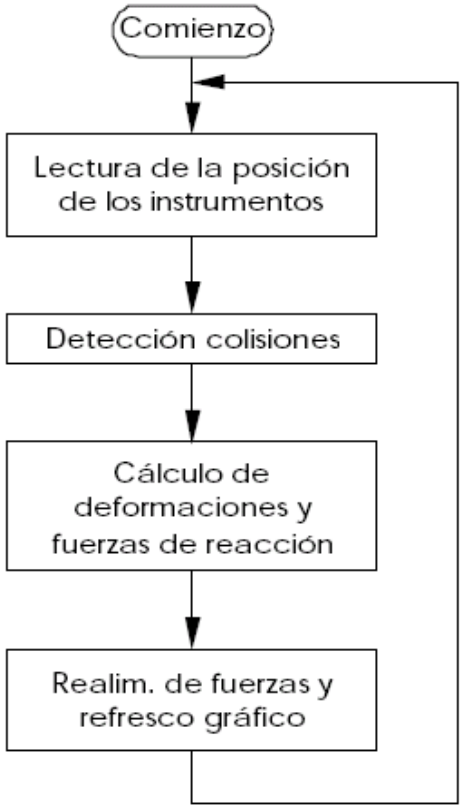


Fig. 1 Ciclo Básico de la Simulación quirúrgica.

Con todo lo mencionado, sin tener en cuenta el costoso cálculo computacional de las deformaciones de los tejidos y las fuerzas de realimentación, cada ciclo de simulación requiere aproximadamente 10 milisegundos. Esto provoca que para que la sensación de tacto y visual sea realista, los correspondientes ciclos de cálculo se han de separar en los simuladores Fig. 2 (Cotin, et al., 1999).

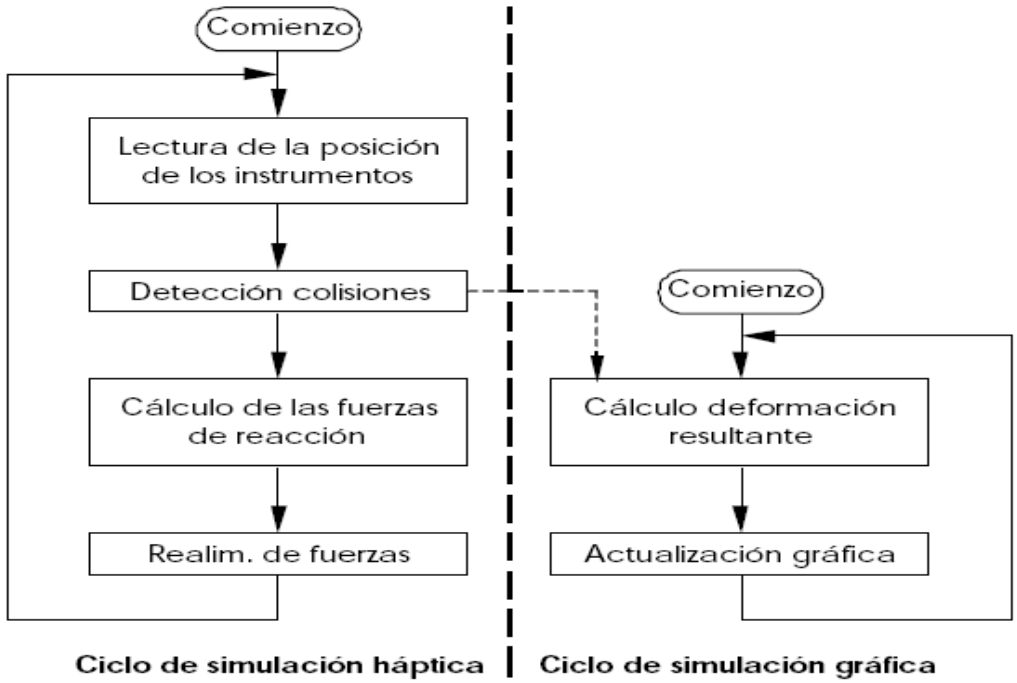


Fig. 2 Ciclos de realimentación visual y de tacto de los SMQ.

1.5 Estado actual del desarrollo de los Simuladores Quirúrgicos en Cuba.

En Cuba no existen Simuladores Quirúrgicos, sin embargo, algunos expertos han desarrollado diferentes métodos de enseñanza para llevar a cabo técnicas de CMA, estas técnicas de entrenamiento son rústicas, pues se realizan utilizando diferentes tipos de frijoles. En el Hospital Clínico-Quirúrgico "Hermanos Ameijeiras" existe un grupo de cirugía *artroscópica* encabezado por el Dr. Carlos Rodríguez Blanco, quien utiliza un laboratorio docente para la instrucción y facilita la adquisición de una sólida base teórico-práctica que disminuye el período de aprendizaje y eleva la competencia del profesional formado.

La UCI cuenta con el Proyecto Productivo SMQ que se dedica a la investigación y desarrollo de Simuladores Quirúrgicos, en estos momentos se está trabajando en la creación de un producto conocido como KHEIPROS, el cual establece la apertura del camino de la RV aplicada a la medicina en Cuba.

1.6 El Simulador KHEIPROS.

El proyecto SMQ de la UCI ofrecerá un simulador para el entrenamiento y la planificación de intervenciones íntegramente cubano. Dicho producto será insertado de manera gradual en el proceso de enseñanza de los procedimientos de CMA en Cuba y podrá ser utilizado de manera independiente por todas las instituciones cubanas de salud, con servicios de CMA, así como instituciones similares de América Latina y el Caribe.

El producto permitirá al usuario ejecutar un conjunto de ejercicios de entrenamiento con diferentes niveles de complejidad que irán aumentando el rigor y el nivel de realismo a medida que el usuario avance en la ejecución, hasta el punto cúspide en el que tendrá la posibilidad de planificar y repasar virtualmente una operación con modelos 3D generados a partir de información real de un paciente obtenido por equipos Tomografía Axial Computarizada (TAC) o Imagen de Resonancia Magnética (MRI).

El mismo poseerá un módulo de registro en el cual los usuarios gestionan sus datos y sus evaluaciones en el entrenamiento. Este producto usará como hardware una computadora personal con dos procesadores, reforzada desde el punto de vista gráfico y una maqueta de sistema quirúrgico conectado a dispositivos diseñados para la determinación de la posición espacial y la realimentación de fuerzas.

El proyecto no cuenta con el personal calificado para su desarrollo, debido a que en su mayoría está conformado por estudiantes de diferentes años que aún no tienen la experiencia suficiente, otro grupo importante que forma parte de él son los profesores recién graduados que no tienen la preparación y el conocimiento que un proyecto de este nivel necesita.

Inicialmente el único cliente con que cuenta el proyecto son las Instituciones Médicas Cubanas que practiquen la CMA, a las cuales el producto se les distribuirá de manera gratuita.

Para cada versión del SMQ KHEIPROS se brindará la documentación y ayuda correspondiente, además se planificarán y ejecutarán cursos de preparación de personal para un mejor entendimiento de los usuarios finales en el uso del simulador.

El precio de comercialización para futuros clientes extranjeros no está definido aún, ya que depende de factores que hasta el momento son irreales.

El producto será multiplataforma por lo que supone la existencia de algún sistema operativo: Windows o GNU/ Linux.

1.6.1 Algunos aspectos de las etapas de desarrollo del software.

El SMQ KHEIPROS ha definido hasta el momento 4 etapas de desarrollo; en este epígrafe se abordarán algunas de las actividades y tareas que se han definido con el fin de lograr un proceso de desarrollo mucho más organizado y acoplado que responde al nivel que exige el proyecto.

Se definieron además los roles participantes por etapa de desarrollo y las responsabilidades que ocupan, a continuación se dan a conocer cada uno de ellos:

- Líder de Proyecto:
  - Define la organización y estructura (líneas de trabajo) del proyecto.
  - Gestiona y asigna recursos humanos y de otro tipo.
  - Establece los horarios de trabajo del proyecto.
  - Establece las estrategias de desarrollo del proyecto.
  - Planifica las fases e iteraciones.
  - Define, planifica, asigna y controla las tareas del proyecto.
  - Coordina las interacciones con los clientes y los usuarios finales.
  - Define el plan de capacitación y evaluación del personal.
  - Planifica y realiza reuniones de información y control del proyecto.
  - Realiza talleres y consejos técnicos con todos los miembros del proyecto.
  - Informa sobre el estado actual del proyecto a los miembros del mismo y a instancias superiores.
  - Motiva y organiza el equipo de trabajo para lograr un objetivo definido.
  - Participa en la selección del personal del proyecto.
  - Participa en los cursos de capacitación para líderes de proyecto.
- Jefe de equipo:
  - Guía al equipo de desarrollo según las estrategias trazadas.
  - Guía al equipo en la especificación del diseño del software.

  - Guía al equipo en las pruebas del sistema.
  - Guía al equipo en la producción de la documentación del usuario.
  - Participa en la producción del reporte de desarrollo.
- Investigador:
  - Investiga sobre las diferentes líneas de investigación definidas en el proyecto.
  - Genera la documentación sobre el estado del arte del tema investigado.
- Arquitecto de Software:
  - Responsable de la arquitectura del software.
  - Decisiones técnicas más importantes en cuanto a las restricciones del diseño global e implementación del proyecto.
- Analista Principal:
  - Define una estrategia para la captura de requisitos.
  - Define los artefactos que se obtendrán como resultado del análisis y la metodología que se sigue para obtenerlos.
  - Define las técnicas de recopilación de información que serán usadas durante la captura de requisitos.
  - Supervisa y controla el cumplimiento de la metodología para el análisis.
  - Define los sistemas, subsistemas y módulos en que se organiza la solución de software.
- Analista de Software:
  - Dirige y coordina el proceso de extracción de requisitos y define las funcionalidades y límites del sistema.
  - Especifica los requerimientos del sistema.
  - Diseña el sistema en cuanto a requisitos y proceso de desarrollo.
  - Asesorar a programadores en cuanto a preparación como analistas de software.
- Programador:
  - Participar en la etapa de diseño junto con los demás especialistas.
  - Desarrollar los componentes, módulos o subsistemas que tenga a cargo.
  - Documentar el código escrito durante el desarrollo de las tareas que se le asignen.
  - Realiza las pruebas de unidad de los componentes.
  - Documenta los juegos de datos utilizados en las pruebas así como los resultados de las mismas.

  - Dar soporte técnico al código realizado por él u otros programadores.
  - Participar en la preparación de los roles de años inferiores para futuros programadores.
- Diseñador 3D:
  - Generación de interfases para los sistemas informáticos.
  - Realización de entornos virtuales.
  - Producción de librerías de medias para la organización y control del desarrollo de los entornos virtuales.
- Diseñador de Interfaz de usuario.
  - Diseña prototipos de interfaz de usuario según los requerimientos de usabilidad del cliente.
  - Encargado de realizar el trabajo artístico que requiera el proyecto (íconos, logos, pantalla de splash, gráficos, etc.)
- Diseñador principal de base de datos.
  - Define el gestor de base de datos a usar.
  - Define la herramienta de modelado para bases de datos relacionales.
  - Define las políticas de cambio sobre los elementos de datos.
  - Define los algoritmos de réplica, sincronización, respaldo y recuperación de la base de datos.
  - Define las políticas de almacenamiento de los datos.
  - Define las políticas de uso de los diferentes objetos de bases de datos ante situaciones particulares.
  - Diseña el modulo de gestión de la Base de Datos.
- Diseñador de casos de prueba.
  - Identificar técnicas apropiadas, herramientas e instrucciones para la implementación de las pruebas.
  - Diseñar los casos de pruebas.
  - Definir listas de chequeo para las pruebas.
- Auditor de la calidad.
  - Elaborar los criterios de la auditoría.
  - Ejecutar las auditorias planificadas.
  - Registrar los resultados de las auditorías.
- Documentador.
  - Elabora lista de chequeo para las revisiones.

  - Revisa todos los artefactos que se generan en el proyecto.
  - Registrar los resultados de las revisiones.
- Planificador de la calidad.
  - Planificar el proceso de Aseguramiento de la Calidad en el proyecto (Plan de Calidad).
  - Realizar el Plan de Prueba, de Revisión y Auditoría de cada iteración.
  - Coordinar el proceso de recopilación, análisis y reporte de las estadísticas de calidad (Plan de mediciones).
  - Hacer seguimiento a estos planes.
  - Guiar las revisiones técnicas formales.
  - Guiar las pruebas que se realicen.
  - Guiar las auditorías que se realicen.
  - Manejar todo lo relacionado con los riesgos de calidad (Lista de Riesgos).
  - Realizar el Resumen de Evaluación de Pruebas.
- Arquitectura y tecnologías.
  - Responsable de la selección, gestión y obtención de las herramientas que se utilizarán en el proyecto.
  - Debe instalar, configurar y asegurar que estas herramientas funcionan como se espera.
- Administrador y configuración.
  - Planificar y chequear el proceso de gestión de configuración (Plan de Gestión de Configuración).
  - Responsable del sistema de gestión de información, salvas, etc.

Las actividades que se definieron para el desarrollo de este software son: entradas, salidas, tiempo de duración y otras no menos importantes, cada una de ellas tiene tareas definidas para que se puedan desarrollar de forma más eficiente.

En la 1ra Etapa:

- **Hito**

Lograr un Simulador de habilidades básicas, a partir de modelos de imitación de procesos. Este Simulador estará conformado por tres ejercicios (Coordinación, Selección y Transferencia y

Focalización de la Cámara), estos están enfocados a la puesta en práctica de algunas de las habilidades básicas que deben alcanzar los cirujanos.

- **Tiempo estimado de duración**

A esta primera etapa se le estima un tiempo de duración de 5 meses.

En la 2da Etapa:

- **Hito**

Producir un Simulador de habilidades avanzadas a partir de la imitación de procesos, incluyendo endoscopía.

- **Tiempo estimado de duración**

Esta etapa tiene un tiempo de duración de 8 meses.

En la 3ra Etapa:

- **Hito**

Elaborar la primera versión del Simulador a partir de la generación de órganos virtuales.

- **Tiempo estimado de duración**

Esta etapa se desarrolla en 7 meses aproximadamente.

En la 4ta Etapa:

- **Hito**

Obtener un Simulador con órganos adquiridos a partir de scanner (TAC, RMI) y retroalimentación al tacto.

- **Tiempo estimado de duración**

Para esta etapa se estima un tiempo de 12 meses.

A lo largo del desarrollo del proyecto SMQ KHEIPROS y con el avance que se vaya obteniendo en cada una de las etapas de desarrollo se pueden presentar diferentes riesgos que se agrupan en: personales, organizativos, de estimación y tecnológicos.

- **Riesgos que se presentan en las etapas de desarrollo**

Los riesgos que a continuación se describen están presentes en cada una de las etapas de desarrollo.

1. Ausencia de tecnología y equipamiento necesario, lo que trae como consecuencia el retraso en el cumplimiento de las tareas planificadas.
2. La falta de experiencia del equipo de trabajo al ejecutar la estrategia del desarrollo del software, provoca obtener resultados de poca calidad o ineficientes, con pérdidas de tiempo por rectificaciones.
3. La falta de comunicación con el gerente del proyecto de SIMPRO retrasa el desarrollo del producto pues no se cuenta con la asesoría requerida y se pierde la relación con el cronograma general.
4. No pactar las tareas y requerimientos con el cliente significa realizar un trabajo sin objetivos claros y no responder a requerimientos específicos del cliente.
5. Realizar estimaciones incorrectas de tiempo y de recursos implica una respuesta fuera de tiempo de los requerimientos o tareas pactadas.
6. Pérdida de información provocando desde pequeños retrasos del trabajo hasta una pérdida importante y total del trabajo realizado.
7. El uso de la tecnología interna con falta de potencial trae consigo el atraso del proyecto en temas complejos, principalmente en cuestiones físicas.
8. La insatisfacción provocada en el cliente compromete la no adquisición de reputación y además desechar el trabajo realizado.
9. La rotura de máquinas acarrea pérdida de la información y de personal sin puesto de trabajo.

Nota: En la cuarta etapa no existen los tipos de riesgos que se plantean en los puntos 2, 3 y 4.

1.7 El proceso de desarrollo del software.

¿Qué es un proceso de desarrollo del software?

Un proceso de desarrollo del software "es aquel en que las necesidades del usuario son traducidas en requerimientos del software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo" (Pressman).

1.7.1 Proceso, métodos y herramientas.

La Ingeniería del software es una tecnología multicapa y como cualquier enfoque de ingeniería debe apoyarse sobre un compromiso de organización de calidad.

El fundamento de la ingeniería del software es la capa de **proceso**. El proceso de la ingeniería del software es la unión de las capas de tecnología y que permite un desarrollo racional y oportuno de la ingeniería del software. El proceso define un marco de trabajo para un conjunto de *áreas clave de proceso* (ACPs) (Paulk, 1993).

Los **métodos** de la ingeniería del software indican cómo construir técnicamente el software. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Los métodos de la ingeniería del software dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelo y otras técnicas descriptivas.

Las **herramientas** de la ingeniería del software proporcionan un enfoque automático o semi-automático para el proceso y para los métodos.

1.7.2 Visión general de la ingeniería de software.

El trabajo que se asocia a la ingeniería del software se puede dividir en tres fases genéricas, independientemente del área de aplicación, tamaño o complejidad del proyecto.

La fase de definición se centra sobre el qué. Es decir, durante la definición, el que desarrolla el software intenta identificar qué información ha de ser procesada, qué función y rendimiento se desea, qué comportamiento del sistema, qué interfaces van a ser establecidas, qué restricciones de diseño existen, y qué criterios de validación se necesitan para definir un sistema correcto. Por tanto, han de identificarse los requisitos claves del sistema y del software. Aunque los métodos durante esta fase irán variando en dependencia del paradigma de ingeniería del software (o combinación de paradigmas) que se aplique, de alguna manera tendrán lugar tres tareas principales: ingeniería de sistemas o de información, planificación del proyecto del software y análisis de los requisitos (Pressman).

La fase de desarrollo se centra en el cómo. Es decir, durante el desarrollo un ingeniero del software intenta definir cómo han de diseñarse las estructuras de datos, cómo ha de implementarse la función dentro de una arquitectura de software, cómo han de implementarse los detalles procedimentales, cómo han de caracterizarse interfaces, cómo ha de traducirse el diseño en un lenguaje de programación (o lenguaje no procedimental) y cómo ha de realizarse la prueba. Los métodos aplicados durante la fase de desarrollo varían, aunque las tres tareas específicas técnicas deberían ocurrir siempre: diseño del software, generación de código y prueba del software (Pressman).

La fase de mantenimiento se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software y a cambios debido a las mejoras producidas por los requisitos cambiantes del cliente (Pressman).

1.7.3 Modelado visual del proceso de desarrollo del software.

Para crear un proceso de desarrollo del software Fig. 3 se establece un marco común del proceso definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos del software, independientemente de su tamaño o complejidad. Un conjunto de tareas conformados por tareas de trabajo de ingeniería del software, hitos de proyectos, productos de trabajo y puntos de aseguramiento de la calidad, que permiten que las actividades del marco de trabajo se adapten a las características del proyecto de software y a los requisitos del equipo del proyecto. Las actividades de gestión de riesgos se realizan independientemente de cualquier actividad del marco de trabajo y aparecen durante todo el proceso (Pressman).

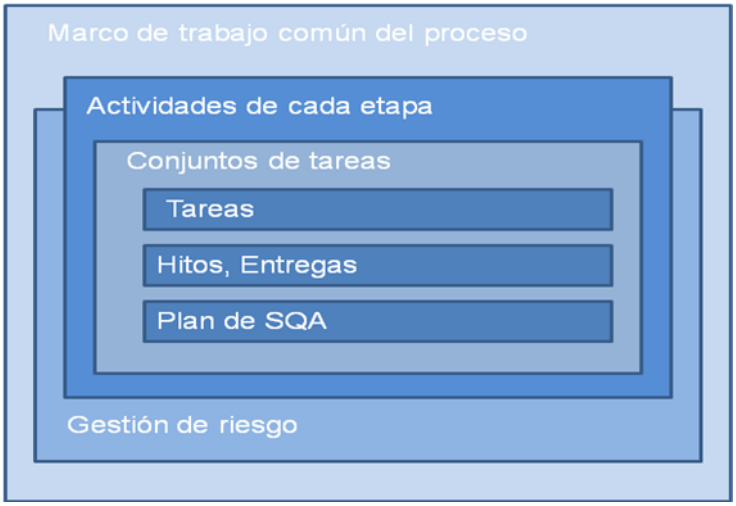


Fig. 3 Proceso del software.

1.7.4 Modelos de Procesos del Software.

Para resolver los problemas reales de una industria u organización, un ingeniero del software o un equipo de ingenieros deben incorporar una estrategia de desarrollo que acompañe al proceso, métodos, capas de herramientas y las fases genéricas. Esta estrategia a menudo se llama modelo de proceso o paradigma de ingeniería del software. Se selecciona un modelo de proceso para la ingeniería del software según la naturaleza del proyecto y de la aplicación, los métodos y las herramientas a utilizarse, los controles y entregas que se requieren.

Todo el desarrollo del software se puede caracterizar como bucle de resolución de problemas en el que se encuentran cuatro etapas distintas: estado actual, definición de problemas, desarrollo técnico e integración de soluciones. Estado actual representa el estado actual de sucesos (Pressman); la definición de problemas identifica el problema específico a resolverse; el desarrollo técnico resuelve el problema a través de la aplicación de alguna tecnología y la integración de soluciones ofrece los resultados (por ejemplo: documentos, programas, datos, nuevo producto) a los que solicitan la solución en primer lugar.

Existen varios modelos de procesos del software, sin embargo se seleccionó como guía a seguir para el proceso de desarrollo del SMQ KHEIPROS de la Facultad 5, el **Modelo incremental** (Pressman), que se encuentra dentro de los **Modelos evolutivos del proceso de software** (Pressman). Estos modelos evolutivos se caracterizan por ser iterativos y permitir que se desarrollen versiones cada vez más completas del software.

El Modelo Incremental combina elementos del modelo lineal secuencial (aplicados repetidamente) con la filosofía interactiva de construcción de prototipos. Este modelo aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Se debería tener en cuenta que el flujo del proceso de cualquier incremento puede incorporar el paradigma de construcción de prototipos Fig. 4.

El modelo incremental entrega el software en partes pequeñas, pero utilizables, llamados incrementos. En general, cada incremento se construye sobre aquel que ya ha sido entregado.

Cuando se utiliza un modelo incremental, el primer incremento a menudo es un producto esencial. Es decir, se afrontan requisitos básicos, pero muchas funciones suplementarias quedan sin extraer. Se analiza la entrega del plan anterior y se desarrolla un plan para el incremento siguiente. El plan afronta la modificación del producto central a fin de cumplir mejor las necesidades del cliente, la entrega de funciones y características adicionales. Este proceso se repite siguiendo la entrega de cada incremento, hasta que se elabore el producto completo.

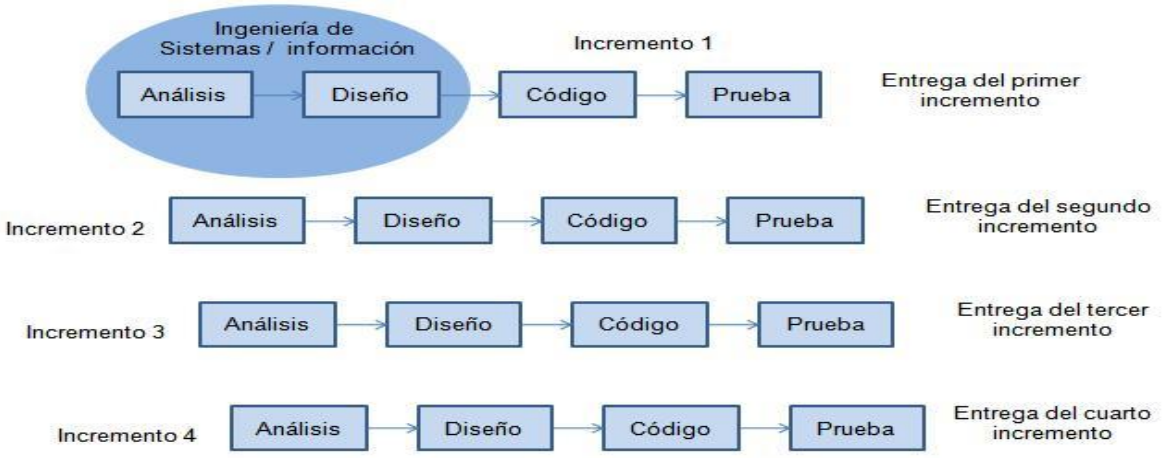


Fig. 4 El modelo incremental.

1.7.5 Tecnología de Procesos.

Las herramientas de tecnología de procesos permiten que una organización de software construya un modelo automatizado del marco de trabajo común de proceso, conjuntos de tareas y gestión de riesgos (Pressman).

Los modelos, presentados normalmente como una red, se pueden analizar para determinar el flujo de trabajo típico y para examinar estructuras alterativas de procesos que pudieran llevar a un tiempo o coste de desarrollo reducidos.

Una vez que se ha creado un proceso aceptable, se pueden utilizar otras herramientas de tecnología de procesos para asignar, supervisar e incluso controlar todas las tareas de ingeniería del software definidas como parte del modelo de proceso. Cada uno de los miembros de un equipo de proyecto de software puede utilizar tales herramientas para desarrollar una lista de control de tareas de trabajo a realizarse, productos de trabajo a producirse y actividades de garantía de la calidad a conducirse. En la Facultad 5, el SMQ utiliza las herramientas STK y TortoiseSVN.

1.8 Análisis de la Norma ISO/IEC 12207.

La estructura de esta norma cubre el ciclo de vida del software desde la conceptualización de ideas hasta la retirada. Adicionalmente, la estructura facilita el control y el mejoramiento de los procesos.

Esta Norma Internacional está diseñada, para ser ajustada a una organización, proyecto o aplicación individual. Establece una estructura común para los procesos del ciclo de vida del software, que contiene procesos, actividades y tareas. Proporciona un proceso que puede emplearse para definir, controlar y mejorar procesos del ciclo de la vida del software. Agrupa las actividades que pueden ser ejecutadas durante el ciclo de vida del software en cinco procesos primarios, ocho procesos de apoyo y cuatro procesos organizacionales Fig. 5. Cada proceso del ciclo de vida está dividido en un conjunto de actividades y cada actividad además está dividida en conjuntos de tareas (Paulk, 1993).

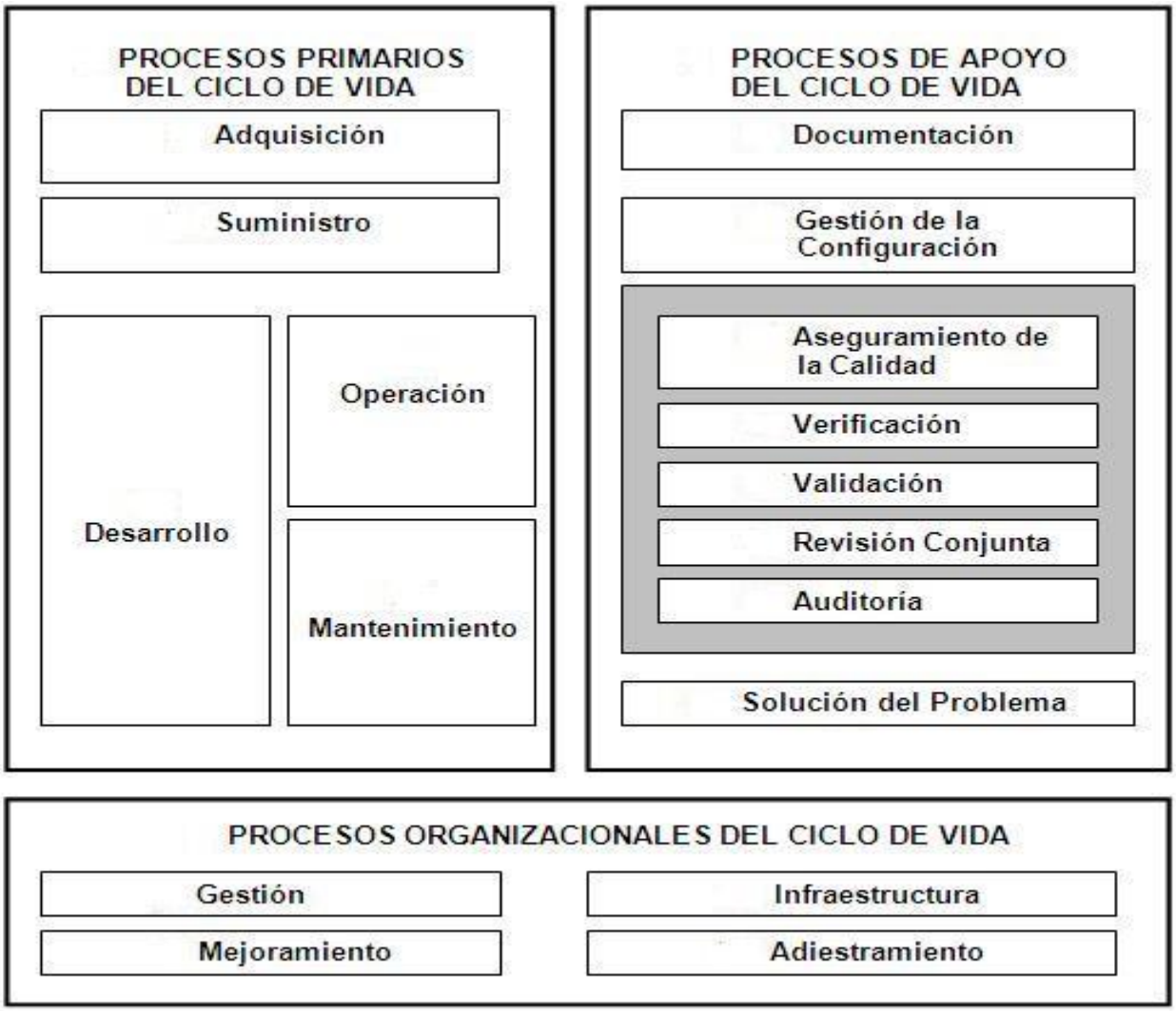


Fig. 5 Estructura de la Norma Internacional ISO/IEC 12207.

Se realizó un análisis exhaustivo de esta norma para utilizarla como guía a la hora de definir el proceso de desarrollo del SMQ de la Facultad 5. Para ello se tuvieron en cuenta fundamentalmente dos puntos, cómo realiza el **Proceso de Desarrollo** y el **Proceso de Mejoramiento**.

1.9.1 Proceso de Desarrollo según la Norma ISO/IEC 12207.

El Proceso de Desarrollo contiene las actividades y tareas de los desarrolladores. El proceso contiene las actividades para el análisis de los requisitos, diseño, codificación, integración, prueba e instalación y aceptación relacionadas con los productos de software.

A continuación se plantean las actividades que se realizan en este proceso:

1) Implementación del proceso.
2) Análisis de los requisitos del sistema.
3) Diseño de la arquitectura del sistema.
4) Análisis de los requisitos del software.
5) Diseño de la arquitectura del software.
6) Diseño detallado del software.
7) Codificación y prueba del software.
8) Integración del software.
9) Prueba de calificación del software.
10) Integración del sistema.
11) Prueba de calificación del sistema.
12) Instalación del software.
13) Apoyo a la aceptación del software.

1.9.2 Proceso de Mejoramiento.

El Proceso de Mejoramiento es un proceso para la implantación, valoración, medición, control y mejoramiento de un proceso del ciclo de vida del software, a continuación se encuentran las actividades que define y las tareas que dan cumplimiento al mismo.

Implantación del proceso

Esta actividad consta de la tarea siguiente:

- La organización deberá implantar un conjunto de procesos organizacionales, para todos los procesos del ciclo de vida del software, de forma similar a como se aplican a sus actividades de negocio. Estos procesos organizacionales y sus aplicaciones a casos específicos se deberán documentar en publicaciones de la organización. Se debería establecer un mecanismo de control del Proceso de Mejoramiento para desarrollar, monitorear, controlar y mejorar el proceso.

Valoración del proceso

Esta actividad consta de las tareas siguientes:

- Un procedimiento para la valoración del proceso que será mejorado debería ser desarrollado, documentado y aplicado. Los registros sobre la valoración deberían ser conservados.
- La organización deberá planificar y realizar revisiones de los procesos en intervalos apropiados para asegurar su continua conformidad y efectividad de acuerdo con los resultados de la valoración.

Mejoramiento del proceso

Esta actividad consta de las tareas siguientes:

- La organización deberá efectuar los mejoramientos a sus procesos organizacionales que determine necesarios como resultado de la valoración y la revisión de un determinado proceso del ciclo de vida del software. La documentación de este proceso debería ser actualizada para reflejar las mejoras en los procesos organizacionales.
- Los datos históricos, técnicos y de las evaluaciones deberían ser recopilados y analizados para obtener un conocimiento de las fortalezas y las debilidades de los procesos del ciclo de vida del software empleados. Estos análisis deberían ser utilizados como retroalimentación para mejorar estos procesos, para recomendar cambios en la dirección de los proyectos (o proyectos subsiguientes) y para determinar las necesidades de modernización de la tecnología.
- Los datos del costo de la calidad deberían ser recopilados, conservados y utilizados para mejorar los procesos de la organización, como una actividad de la gestión. Estos datos deberán servir al propósito del establecimiento del costo, tanto de la prevención y de la solución de los problemas como de las no conformidades, de los productos y servicios del software.

CAPÍTULO 2. ANÁLISIS DE LAS METODOLOGÍAS DE DESARROLLO DEL SOFTWARE

Este Capítulo estará dedicado al estudio de diferentes metodologías, debido a que son un eslabón fundamental y por tanto de gran utilidad a la hora de guiar la estructura de cualquier proyecto de software; actualmente existen diversas metodologías, aunque este número se reduce al enmarcarnos solo a proyectos de RV, de ahí que para realizar este estudio y siguiendo criterios de expertos solo cuatro de ellas fueron seleccionadas y se clasifican en Metodologías Tradicionales y Metodologías Ágiles, dentro de la primera clasificación se encuentra el Proceso Unificado Racional (RUP), que es muy importante estudiar, pues es por la que actualmente se está rigiendo el proyecto SMQ. Por otra parte están la Programación Extrema (XP), Marco de la solución de Microsoft (MSF) y el Proceso Unificado Básico (BUP) que se encuentran dentro de las ágiles. Todo este análisis tiene como fin escoger características favorables de cada metodología que permitan posteriormente definir un proceso de desarrollo completo y mucho más óptimo.

¿Qué es una metodología de desarrollo de software?

Se conoce como metodología de desarrollo de software al conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software con el objetivo de formalizarlo y optimizarlo.

Es aquella rama dentro de la Ingeniería de Software que se encarga de elaborar estrategias de desarrollo del software que promuevan prácticas adaptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente.

Una metodología de desarrollo del software representa el camino a seguir para desarrollar software de manera sistemática.

2.1 Rasgos generales de las metodologías de desarrollo de software.

Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el proceso indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales Fig. 6.

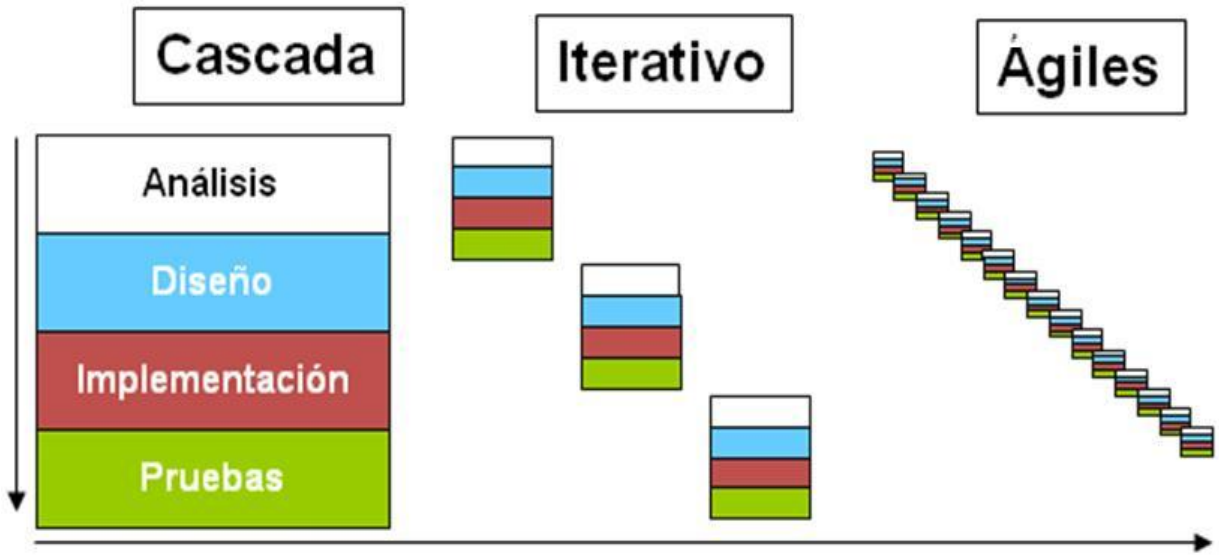


Fig. 6 Ciclo de vida y modelos de las metodologías.

La finalidad de una metodología de desarrollo es garantizar la eficacia (cumplir con los requisitos iniciales) y la eficiencia (minimizar las pérdidas de tiempo) en el proceso de generación de software.

Algunas tareas complementarias que se realizan a lo largo de todo el ciclo de desarrollo de software, son:

- Gestión de la configuración: identificación de versiones, control de cambios, etc.
- Gestión de la calidad: seguimiento de errores, revisiones del nivel de calidad.
- Revisión de las premisas iniciales: revisión de los requerimientos y de los diseños.
- Gestión del entorno de desarrollo: herramientas de desarrollo, ficheros, recursos, etc.

Características que debe tener toda metodología

La utilización de metodologías es algo que no puede faltar a la hora de llevar a cabo un proyecto, puesto que facilita la organización y comprensión del mismo, por lo que es recomendable que cuente con:

- La existencia de un conjunto de reglas predefinidas.
- Una cobertura total del ciclo de desarrollo.
- Hacer verificaciones intermedias.

- Realizar planificación y control durante todo el desarrollo del proyecto.
- Desarrollar una comunicación efectiva.
- Fácil formación.
- Realizar actividades que mejoren el proceso de desarrollo.
- Un soporte de mantenimiento.
- Un buen soporte de la reutilización de software.

2.2 Metodologías tradicionales de desarrollo del software.

Se caracterizan por exponer procesos basados en planeación exhaustiva. Esta planeación se realiza esperando que el resultado de cada proceso sea determinante y predecible. La experiencia ha mostrado que, como consecuencia de las características del software, los resultados de los procesos no son siempre predecibles y sobre todo, es difícil predecir desde el comienzo del proyecto cada resultado.

Dentro de las características más importantes de las metodologías tradicionales se tiene que:

- Están enfocadas al proceso definido por la metodología y no a las características del proyecto.
- Establecen fases rígidas.
- Cada fase tiene una serie de documentos sumamente explícitos.
- Se basan en la filosofía basada en predicción del proyecto.

Dentro de las metodologías tradicionales más utilizadas para el desarrollo de este tipo de software se encuentra RUP.

2.2.1 Proceso Unificado Racional (RUP).

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, es un proceso de desarrollo de software. Un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación. Se basa en la unión de pequeños componentes. RUP es un proceso para el desarrollo de un producto que define claramente quién, cómo, cuándo y qué debe hacerse en el proyecto.

RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Como RUP es un proceso, en su modelación define como sus principales elementos:
Trabajadores: define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
Actividades: es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
Artefactos: productos tangibles del proyecto que son elaborados, modificados y usados en las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
Flujo de actividades: secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

El **ciclo de vida** de RUP se caracteriza por ser:

Dirigido por casos de uso: los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan cómo se llevan a cabo los casos de uso.

Centrado en la arquitectura: la arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de Lenguaje de Modelado Unificado (UML).

Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos al

crecimiento del producto. Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierta 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.

Algunas de las ventajas que tiene es que las pruebas continuas e iterativas promueven una mejor evaluación del estado del proyecto. Los patrocinadores reciben evidencia concreta del avance del proyecto. Se pueden acomodar mejor los cambios (requerimientos, tácticos y tecnológicos). Los problemas más complejos se atacan primero.

A lo largo del proceso de desarrollo de RUP se definen una serie de **principios claves** para lograr una mejor organización durante el ciclo de vida del proyecto, estos son:

Adaptación del proceso: el proceso deberá adaptarse a las características propias de la organización. El tamaño del mismo, así como las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.
Balancear prioridades: los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.
Colaboración entre equipos: el desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requerimientos, desarrollo, evaluaciones, planes, resultados, etc.
Demostrar valor iterativamente: los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.
Elevar el nivel de abstracción: este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software o esquemas (*frameworks*) por nombrar algunos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.
Enfocarse en la calidad: el control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción.

Flujos de trabajo y descripción de las actividades por fases.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los 3 últimos como flujos de apoyo. Divide el proceso de desarrollo en ciclos, cada uno de ellos se divide en cuatro fases dentro de las cuales se realizan varias iteraciones, la cantidad de iteraciones varía según la complejidad del proyecto, unas tienen más peso que otras en dependencia de las actividades a desarrollar, obteniendo un producto final al culminar cada ciclo que finaliza con un hito:

Fase de Inicio:

Durante la fase de inicio las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de levantamiento de requisitos.

Modelado del negocio: En esta actividad el equipo se familiarizará más con el funcionamiento de la empresa. Los objetivos que persigue esta actividad son (2002):

- Entender la estructura y la dinámica de la organización para la cual el sistema va a ser desarrollado.
- Entender el problema actual en la organización e identificar mejoras potenciales.
- Asegurar que clientes, usuarios finales y desarrolladores tengan un entendimiento común de la organización.

Levantamiento de Requisitos: Este es uno de los flujos de trabajo más importantes, porque en él se establece qué es lo que tiene que hacer exactamente el sistema que se va a construir. A continuación se muestran los objetivos de este flujo (2002):

- Establecer y mantener un acuerdo entre clientes y otros desarrolladores sobre lo que el sistema podría hacer.
- Proveer a los desarrolladores un mejor entendimiento de los requisitos del sistema.
- Definir el ámbito del sistema.
- Proveer una base para estimar costos y tiempo de desarrollo del sistema.
- Definir una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario.

Objetivos de la fase de Inicio (Kruchten, 2000)**:**

- Establecer el ámbito del proyecto y sus límites.
- Encontrar los casos de uso críticos del sistema y los escenarios básicos que definen la funcionalidad.
- Mostrar al menos una arquitectura candidata para los escenarios principales.
- Estimar el coste en recursos y tiempo de todo el proyecto.
- Estimar los riesgos y las fuentes de incertidumbre.
- Hacer levantamiento de requisitos y el modelado del negocio.

Resultados de la fase de Inicio (1998)**:**

- Tener elaborado el documento visión, donde se expone una visión general de los requerimientos del proyecto, características claves y restricciones principales.
- El modelo inicial de casos de uso debe tener completado de un 10% a un 20% como mínimo.
- El glosario inicial que no es más que la terminología clave del dominio.
- Lista de riesgos y plan de contingencia.
- Plan del proyecto elaborado, mostrando fases e iteraciones.
- Modelo de negocio definido, si es necesario.
- Obtención de prototipos exploratorios para probar conceptos o la arquitectura candidata.

Si durante la planificación del proyecto en cuestión los expertos o los clientes se percatan que este no es viable y por tanto que no cumplirá con las metas esperadas, entonces se decide abandonar el proyecto.

Fase de Elaboración:

En la fase de elaboración, las iteraciones se orientan al desarrollo de la línea base de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la línea base de la arquitectura.

Flujo de Análisis y Diseño: en él se especifican los requerimientos y se describen sobre cómo se van a implementar en el sistema. Y comprende las siguientes actividades (2002):

- Transformar los requisitos al diseño del sistema.
- Desarrollar una arquitectura para el sistema.

- Adaptar el diseño para que sea consistente con el entorno de implementación.

Objetivos de la fase de Elaboración (Kruchten, 2000)**:**

- Definir, validar y cimentar la arquitectura.
- Completar la visión.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones y debe incluir los costes.
- Demostrar que la arquitectura propuesta soportará la visión con un coste y en un tiempo razonable.

Resultados en la fase de Elaboración (1998)**:**

- Un modelo de casos de uso completo al menos hasta el 80%: todos los casos de uso y actores del sistema identificados, la mayoría de los casos desarrollados.
- Requisitos adicionales que capturan los requisitos no funcionales y cualquier requisito no asociado con un caso de uso específico.
- Descripción de la arquitectura de software.
- Mostrar un prototipo ejecutable de la arquitectura.
- Tener definida una lista de riesgos refinada.
- Plan de desarrollo para el proyecto.
- Un manual de usuario preliminar (opcional).

Fase de Construcción:

En esta fase se profundiza en el diseño de los componentes y de manera iterativa se van añadiendo las funcionalidades al software a medida que se construyen y prueban, permitiendo a la vez que se puedan ir incorporando cambios. De esta forma se puede construir el producto, evolucionar la visión, la arquitectura y los planes hasta obtener el producto listo para ser entregado a la comunidad de usuarios. El énfasis está en la producción eficiente y no en la creación intelectual. Incluye los flujos de trabajo de Implementación, prueba y despliegue.

Implementación: se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. El resultado final es un sistema ejecutable. Y comprende las siguientes actividades.

- Planificar qué subsistemas deben ser implementados y en qué orden deben ser integrados, formando el Plan de Integración.
- Cada desarrollador decide en qué orden implementa los elementos del subsistema.
- Si encuentra errores de diseño, los notifica.
- Se integra el sistema siguiendo el plan.

Pruebas: Este flujo de trabajo es el encargado de evaluar la calidad del producto que se está desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida. Este flujo abarca las actividades siguientes (2002):

- Encontrar y documentar defectos en la calidad del software.
- Generalmente asesora sobre la calidad percibida del software.
- Proveer la validación realizada en el diseño y la especificación de requisitos por medio de demostraciones concretas.
- Verificar las funciones del producto de software según lo diseñado.
- Verificar que los requisitos tengan su apropiada implementación.

Despliegue: Esta actividad tiene como objetivo producir con éxito distribuciones del producto y distribuirlo a los usuarios. Las actividades implicadas son:

- Probar el producto en su entorno de ejecución final.
- Empaquetar el software para su distribución.
- Distribuir el software.
- Instalar el software.
- Proveer asistencia y ayuda a los usuarios.
- Formar a los usuarios y al cuerpo de ventas.
- Migrar el software existente o convertir bases de datos.

Objetivos de la fase de Construcción (Kruchten, 2000)**:**

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- Conseguir una calidad adecuada tan rápido como sea práctico.

Resultados en la fase de Construcción (1998)**:**

- Modelos completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación)
- Arquitectura íntegra (mantenida y mínimamente actualizada)
- Riesgos presentados mitigados.
- Plan del proyecto para la fase de Transición.
- Manual inicial de usuario

Criterios de evaluación en esta fase:

- El producto obtenido es estable y maduro como para ser entregado a la comunidad de usuario para ser probado.
- Todos los usuarios expertos están listos para la transición en la comunidad de usuarios.

Fase de Transición:

Esta fase se ocupa del traslado del software desde los entornos de desarrollo a los entornos de producción, en los que el usuario final hará uso del sistema. Esto incluye manufacturarlo, entregarlo, entrenamientos, soporte y mantenimiento del producto hasta que el usuario esté satisfecho.

Algunos objetivos de la fase de Transición:

- Conseguir que el usuario se valga por sí mismo para manipular el software.
- Contar con un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario.

Resultados en la fase deTransición (1998)**:**

- Prototipo operacional.
- Documentos legales.
- Línea de base del producto completa y corregida que incluye todos los modelos del sistema.
- Descripción de la arquitectura completa y corregida.
- Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión.

Criterios de evaluación en esta fase:

- El usuario se encuentra satisfecho.
- Son aceptables los gastos actuales versus los gastos planificados

A lo largo de todo el desarrollo del proyecto se deben realizar actividades como:

Administración del proyecto: Se vigila el cumplimiento de los objetivos, gestión de riesgos y restricciones para desarrollar un producto que esté acorde a los requisitos de los clientes y los usuarios.

- Proveer un marco de trabajo para la gestión de proyectos de software intensivos.
- Proveer guías prácticas, realizar planeación, contratar personal, ejecutar y monitorear el proyecto.
- Proveer un marco de trabajo para gestionar riesgos.

Administración de configuración y cambios: El control de cambios permite mantener la integridad de todos los artefactos que se crean en el proceso, así como de mantener información del proceso evolutivo que han seguido.

Ambiente: La finalidad de esta actividad es dar soporte al proyecto con las adecuadas herramientas, procesos y métodos. Brinda una especificación de las herramientas que se van a necesitar en cada momento, así como definir la instancia concreta del proceso que se va a seguir.

Las responsabilidades de este flujo de trabajo incluyen:

- Selección y adquisición de herramientas.
- Establecer y configurar las herramientas para que se ajusten a la organización.
- Configuración del proceso.
- Mejora del proceso.
- Servicios técnicos.

Roles definidos por RUP

RUP es un proceso que define diferentes roles para cada fase, algunos de ellos están presentes en todo o casi todo el transcurso del proyecto según las actividades que correspondan realizar en cada etapa. A continuación se mencionan y desglosan según su clasificación(2002):

Analistas:

- Analista de procesos de negocio.
- Diseñador del negocio.
- Analista de sistema.

- Especificador de requisitos.

Desarrolladores:

- Arquitecto de software.
- Diseñador.
- Diseñador de interfaz de usuario.
- Diseñador de cápsulas.
- Diseñador de base de datos.
- Implementador.
- Integrador.

Gestores:

- Jefe de proyecto.
- Jefe de control de cambios.
- Jefe de configuración.
- Jefe de pruebas.
- Jefe de despliegue.
- Ingeniero de procesos.
- Revisor de gestión del proyecto.
- Gestor de pruebas.

Apoyo:

- Documentador técnico.
- Administrador de sistema.
- Especialista en herramientas.
- Desarrollador de cursos.
- Artista gráfico.

Especialista en pruebas:

- Especialista en Pruebas.
- Analista de pruebas.
- Diseñador de pruebas.

Puntos claves de RUP

- Pesado.

- El proceso se divide en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto en las que se hace un mayor o menor énfasis en las distintas actividades según se requiera.
- El discurrir del proyecto se define en flujos de trabajo.
- Es importante que se evalúe la calidad de los artefactos en varios puntos durante el proceso de desarrollo, especialmente al final de cada iteración.
- Está basado en roles.
- Realiza modelado visual usando UML.
- Trabaja de manera muy organizada.
- Garantiza una amplia documentación.

Ventajas de RUP

- Evaluación en cada fase que permite cambios de objetivos.
- Funciona bien en proyectos de innovación.
- Es sencillo, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software.
- Seguimiento detallado en cada una de las fases.

Desventajas de RUP

- La evaluación de riesgos es compleja.
- Excesiva flexibilidad para algunos proyectos.
- Se pone al cliente en una situación que puede ser muy incómoda para él.
- El cliente deberá ser capaz de describir y entender a un gran nivel de detalle para poder acordar un alcance del proyecto con él.

2.3 Metodologías Ágiles de desarrollo del software.

En febrero de 2001, después de celebrada la reunión en Utah-EEUU, nace el término ágil aplicado al desarrollo de software. En esta reunión participan un grupo de expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una

alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Las metodologías ágiles han logrado una gran demanda debido a las ventajas que reportan a lo largo de todo el proceso de desarrollo del software, de cierto modo han revolucionado la manera de producir software, y a la vez generado un amplio debate entre sus seguidores, incluso por quienes no las ven como alternativa para las metodologías tradicionales. Las metodologías ágiles emplean una gran cantidad de mejores prácticas que van desde el diseño del espacio de trabajo del equipo del proyecto, hasta la forma en que se deben establecer reuniones de avance y la recopilación de requerimientos.

Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciendo así los costos de implantación en un equipo de desarrollo.

En el llamado **Manifiesto Ágil**, se resume la filosofía ágil donde se hacen las siguientes valoraciones:

Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las Herramientas: la gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que este configure su propio entorno de desarrollo en base a sus necesidades.

Desarrollar software que funciona más que conseguir una buena documentación: la regla a seguir es “no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante”. Estos documentos deben ser cortos y centrarse en lo fundamental.

La colaboración con el cliente más que la negociación de un contrato: se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos debe ser la que marque la marcha del proyecto y asegure su éxito.

Responder a los cambios más que seguir estrictamente un plan: la habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) Determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

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.

Los principios de las metodologías ágiles son:

- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
- Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
- Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
- Las personas del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
- Construir el proyecto en torno a individuos motivados. Brindarles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- El software que funciona es la medida principal de progreso.
- Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
- La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
- La simplicidad es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Desventajas de las Metodologías Ágiles

- Están dirigidas a equipos pequeños o medianos.
- El entorno físico debe ser un ambiente que permita la comunicación y colaboración entre todos los miembros del equipo durante todo el tiempo.

- Cualquier resistencia del cliente o del equipo de desarrollo hacia las prácticas y principios puede llevar al proceso al fracaso (el clima de trabajo, la colaboración y la relación contractual son claves).
- El uso de tecnologías que no tengan un ciclo rápido de realimentación o que no soporten fácilmente el cambio.

2.3.1 Programación Extrema (XP).

Las raíces de la XP yacen en la comunidad de Smalltalk, y en particular de la colaboración cercana de Kent Beck y Ward Cunningham a finales de los 1980s. Ambos refinaron sus prácticas en numerosos proyectos a principios de los 90s, extendiendo sus ideas de un desarrollo de software adaptable y orientado a las personas.

XP surgió como contrapartida a las metodologías clásicas o tradicionales, añadiendo una nueva forma de hacer las cosas. Centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promueve el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo, se basa en realimentación continua entre el cliente y el equipo de desarrollo, propicia comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

Esta metodología consta de 4 valores imprescindibles para su desarrollo, los mismos se detallan a continuación:

- **Comunicación**: los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.
- **Simplicidad**: codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se requiere mantenimiento o ampliación resulta imposible hacerlo y se tienen que desechar y partir de cero.
- **Realimentación**: mediante la realimentación se ofrece al cliente la posibilidad de conseguir un sistema adecuado a sus necesidades. Se le va mostrando el proyecto a tiempo para sugerir cambios y poder retroceder a una fase anterior para rediseñarlo a su gusto.

- **Coraje**: se debe ser tenaz para cumplir los tres puntos anteriores. Hay que tener valor para comunicarse con el cliente y enfatizar algunos puntos a pesar de que esto pueda dar sensación de ignorancia por parte del programador; hay que ser decidido para mantener un diseño simple y no optar por lo que pudiera parecer mejor o un camino más fácil y por último hay que enfatizar que la realimentación será efectiva.

XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. XP construye un proceso de diseño evolutivo que se basa en refactorizar un sistema simple en cada iteración. Todo el diseño se centra en la iteración actual y no se hace nada anticipadamente para necesidades futuras. El resultado es un proceso de diseño disciplinado, lo que es más, combina la disciplina con la adaptabilidad.

Los principios claves alrededor de los cuales se fundamenta la metodología XP son:

- Acortar los ciclos de desarrollo.
- Involucrar al cliente desde el principio hasta el final de cada ciclo.

Las técnicas de trabajo que proporciona XP consiguen minimizar el impacto que los cambios suponen en un proyecto de desarrollo de software.
Acortar los ciclos de desarrollo y reforzar la comunicación con el cliente le permiten:

- Centrarse cada vez en un problema muy concreto y en el momento justo.
- Solucionarlo de manera consensuada, inmediata y no arrastrarlo a lo largo del proyecto.
- Comenzar cada nuevo ciclo de desarrollo sobre una versión intermedia contrastada, verificada y aceptada por el cliente.

Elementos de XP

Las Historias de Usuario: se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Es una técnica utilizada para especificar los requisitos del software. El tratamiento es muy dinámico y flexible. Las historias de usuario son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración.

Estas tarjetas tienen en su contenido: fecha, tipo de actividad (nueva, corrección y mejora), prueba funcional, número de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción, notas y una lista de seguimiento con la fecha, estado, cosas por terminar y comentarios. A efectos de planificación, las historias pueden ser de una a tres semanas de tiempo de programación (para no superar el tamaño de una iteración) (Jeffries, y otros, 2000).

Roles definidos por XP

Programador: escribe las pruebas unitarias y produce el código del sistema.
Cliente: escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
Encargado de pruebas: ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
Encargado de seguimiento: proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
Entrenador*:* es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
Consultor: es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
Gestor*:* es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Proceso de desarrollo de XP

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos (Jeffries, y otros, 2000):

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.

5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP abarca las siguientes fases (Beck):

- **Exploración:** los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Esta fase toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.
- **Planificación de la Entrega:** el cliente establece la prioridad de cada historia de usuario y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Esta fase dura unos pocos días.
- **Iteraciones:** esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.
- **Producción:** esta fase requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.
- **Mantenimiento:** mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente.

- **Muerte del Proyecto:** el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Prácticas de XP

XP apuesta por un crecimiento lento del costo del cambio y con un comportamiento asintótico. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación de las prácticas que se describen a continuación:

El juego de la planificación: hay una comunicación frecuente entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.

Entregas pequeñas: producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más de 3 meses.

Metáfora: el sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (Fowler, y otros, 2001)(conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).

Diseño simple: se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.

La XP define un "diseño tan simple como sea posible" como aquél que:

- Pasa todas las pruebas.
- No contiene código duplicado.
- Deja clara la intención de los programadores (enfatiza el qué, no el cómo) en cada línea de código.
- Contiene el menor número posible de clases y métodos.

Pruebas: la producción de código está dirigida por las pruebas unitarias. Estas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.

Refactorización: es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo (08Ju). Uno de los objetivos de la XP es mantener la curva de costes tan plana como sea posible, por lo que existen una serie de mecanismos destinados a mantener el código en buen estado, modificándolo activamente para que conserve claridad y sencillez. A este proceso básico para mantener el código en buena forma se le llama refactorización.

Programación en parejas: toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores) (Fowler, et al., 2001).

Propiedad colectiva del código: cualquier programador puede cambiar cualquier parte del código en cualquier momento. Cada cual es responsable de las modificaciones que haga. El principio básico es "quien lo rompe, lo arregla, no importa si está en el código propio o en el de otro". Hay que tener en cuenta que la existencia de pruebas automatizadas impide que se produzca un desarrollo anárquico, al ser cada persona responsable de que todas las pruebas se ejecuten con éxito al incorporar los cambios que ha introducido al programa.

Integración continua: cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día (Fowler, et al., 2001).

40 horas por semana: la XP lleva un modo de trabajo en que el equipo está siempre al 100%. Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo.

Cliente en el equipo: el cliente tiene que estar presente y disponible todo el tiempo para el equipo. Este es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el

trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita (Jeffries, y otros, 2000).

Estándares de programación: XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

El mayor beneficio de estas prácticas se adquiere con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras.

¿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.

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del Sistema.
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

Puntos claves de XP

- Ligero.
- Próximo al desarrollo.
- Se basa en historias de usuario para describir las características que el sistema debe poseer.
- Fuerte comunicación con el cliente.
- El código fuente pertenece a todos.
- La programación se realiza en parejas.
- Las pruebas son la base de la funcionalidad.
- Solo el mínimo de información.

- Escaza documentación.

Aspectos positivos de XP

- Usa procedimientos exhaustivos de pruebas.
- Es ligera, orientada al cliente y de iteraciones cortas y rápidas. No ponen demasiadas tareas organizativas sobre los desarrolladores (como modelado, generación de documentación externa, etc.), minimizan el efecto de esto con un representante del cliente.
- El cliente recibe después de cada iteración un pedazo funcional del programa.
- Si se requieren cambios se ajustará el plan de iteraciones y el de *reléase* y tomará la nueva dirección en el desarrollo.
- Centrada para proyectos cortos y equipos pequeños.
- Debido a que los *reléase* se producen a menudo, se afirma la necesidad de que se puedan realizar pruebas automáticas.
- En XP desde que se inicia el proyecto ya se sabe la cantidad de programadores que van a participar y trabaja en base a facilitar el trabajo de estos, es por ello que su proceso de desarrollo se define casi por completo, incluyendo la parte de prueba e integración del software.
- La implementación de las pruebas antes que la propia funcionalidad hace que el desarrollador tenga que pensar pronto sobre lo que tiene que hacer y cómo probarlo correctamente.
- Se basa en un proceso iterativo. Esto permite acercarse poco a poco a la solución sin entrar demasiado rápido en detalles.
- En XP el desarrollo ve en qué dirección debe ir gracias a la realimentación del cliente, sin ningún tipo de restricción previa.
- Presenta la compartición del código fuente, pues está pensado para equipos pequeños.

Aspectos negativos de XP

- El cliente está en el lugar del desarrollo.
- El aseguramiento de la calidad no se basa en la documentación, si no en controles propios y una comunicación fluida con el cliente.
- Programación en pares.
- La rotación de los miembros del equipo sobre los componentes y emparejamientos.
- La evaluación del estado del proyecto y los reportes, recaen solamente en los jefes de proyecto.

- No requiere ninguna herramienta fuera del ambiente de programación y prueba.

2.3.2 Marco de la solución de Microsoft (MSF).

MSF versión 4.0 es un *metamodelo* para describir el ciclo de vida de desarrollo de software, una flexible e interrelacionada serie de conceptos, modelos y buenas prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. Tiene como ventaja mayor vinculación con el cliente y está orientado al trabajo en equipo.

MSF se puede utilizar por sí mismo o con otras herramientas y técnicas como el Proceso Rational para planear, construir y administrar el desarrollo de soluciones de negocio a la medida.

El resultado de utilizar MSF para el correcto desarrollo de proyectos es mejorar la eficiencia, la calidad de las soluciones entregadas y optimizar el desempeño del equipo de trabajo involucrado, para lograr este objetivo MSF se centra en los modelos de proceso y de equipo, además de las 3 disciplinas:

Componentes de MSF

Esta metodología está compuesta por una serie de principios, disciplinas y modelos que a continuación se detallan (08Ab):

Principios

- Promover comunicaciones abiertas. (comunicación con el cliente)
- Trabajar para una visión compartida.
- Fortalecer los miembros del equipo. (capacitación de las personas)
- Establecer responsabilidades claras y compartidas (incluyendo al cliente).
- Focalizarse en agregar valor al negocio.
- Permanecer ágil y esperar los cambios.
- Invertir en calidad (tiempo, trabajo, dinero).
- Aprender de todas las experiencias.
- Asociado con clientes.
- Siempre crear productos entregables.

Disciplinas: son las actividades que persiguen un objetivo coherente con la filosofía de MSF. Son transversales a los roles. Ellas son (08Ab):

Gestión de Proyectos

Es una disciplina que describe el rol de la gestión del proyecto dentro del Modelo de Equipo de MSF y permite mayor escalabilidad, desde proyectos pequeños a proyectos largos y complejos.

Se basa en:

- Planificar sobre entregas cortas.
- Incorporar nuevas características sucesivamente.
- Identificar cambios ajustando el cronograma.

Control de riesgo

Diseñada para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.

Control de cambios

Diseñada para que el equipo sea proactivo en lugar de reactivo. Los cambios deben considerarse riesgos inherentes y además deben registrarse y hacerse evidentes.

Modelos

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión de Riesgo, Modelo de Diseño de Proceso y Modelo de Aplicación (08Ab).

Modelo de Arquitectura del Proyecto: diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.

Modelo de Equipo: este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo que requiere habilidades y conocimientos especiales en cada etapa. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles. Muestra cómo estructurar equipos de alto desempeño para crear soluciones más rápido, mejor y más baratas. Hace énfasis en las

comunicaciones claras y en un equipo de iguales con papeles y responsabilidades claras en todo el proyecto. La calidad del producto se asegura por cada miembro del equipo. Posibilita que una persona pueda participar en los siguientes roles (08Ab):

- Analista de negocios: crea visión del proyecto, escenarios y requisitos de calidad de servicio.
- Jefe de proyecto: dirige el proyecto, planea y dirige cada iteración.
- Arquitecto: crea la arquitectura de la solución.
- Desarrollador: su única responsabilidad es lanzar el producto.
- Personal de pruebas: cerrar un error y comprobar un escenario.

Modelo de Proceso: diseñado para mejorar el control del proyecto, minimizando el riesgo y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de Equipo. Combina los mejores principios del modelo en cascada y del modelo en espiral, es decir, combina la claridad que planea el modelo en cascada y las ventajas de los puntos de transición del modelo en espiral.

Modelo de Gestión del Riesgo: diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.

Modelo de Diseño de Proceso: diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.

Modelo de Aplicación: diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores.

Modelo de Roles: enfocado a la estructuración de las personas y sus actividades para facilitar el éxito de un proyecto. Define bloques de roles, áreas funcionales, responsabilidades, y dirección para los miembros del equipo. Presenta metas únicas para los roles dentro del ciclo de vida del proyecto. Fomenta la combinación de distintas ideas, a través de equipos de pares. Define roles y responsabilidades para los equipos de pares. Este modelo es el aspecto más distintivo de MSF.

La visión de MSF es proveer una guía de procesos, que sea:
Productiva: soportada por medio de listas de verificación, directrices, en lugar de contenido detallado.
Integrada: gracias a Visual Studio 2005 (VS2005) permite integración de tareas y herramientas.
Extensible: los procesos y guías son adaptables, permitiendo a sus usuarios tomar un enfoque formal o ágil.

Características de MSF:
Adaptable: su uso es limitado a un lugar específico, aunque puede ser usado en cualquier parte.
Escalable: puede organizar equipos tan pequeños entre 3 ó 4 personas, así como también, proyectos que requieren de 50 personas a más.
Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.
Tecnología Agnóstica: puede ser usada para desarrollar soluciones basadas en cualquier tecnología.

Objetivos y entregables de cada fase del proceso de desarrollo del software.
El modelo de procesos de MSF provee una estructura para el desarrollo de aplicaciones que consiste en 4 etapas distintas, cada una de las cuales culmina con una meta definida. Estas etapas son (08Ab1):

Visión (visión y alcance aprobados)
Esta fase trata uno de los requisitos más fundamentales para el éxito del proyecto, la unificación del equipo detrás de una visión común. El equipo debe tener una visión clara de lo que quisiera lograr para el cliente y ser capaz de indicarlo en términos que motivarán a todo el equipo y al cliente. Se definen los líderes y responsables del proyecto, adicionalmente se identifican las metas y objetivos a alcanzar; estas últimas se deben respetar durante la ejecución del proyecto en su totalidad, y se realiza la evaluación inicial de riesgos del proyecto

Objetivo:

Obtener una visión del proyecto compartida, comunicada, entendida y alineada con los objetivos del negocio. Además identificar los beneficios, requerimientos funcionales, sus alcances, restricciones y los riesgos inherentes al proceso.

Entregables:

- Documento visión:
  - Antecedentes y visión.
  - Criterios de diseño.
- Documento detalle de la visión:
  - Beneficios, metas, objetivos y restricciones.
  - Perfiles de usuarios.
  - Casos de uso.
  - Requerimientos funcionales y no funcionales.
  - Requerimientos del sistema.
  - Plan de instalación.
  - Arquitectura lógica (diagrama de componentes UML).
  - Arquitectura física (diagrama de despliegue UML).
- Documento requerimientos funcionales:
  - Descripción detallada de los requerimientos y características que componen cada caso de uso descrito en el documento Detalle de la Visión; indicando perfiles asociados, recursos del equipo de proyecto, riesgos, observaciones y script de pruebas.
- Documento matriz de riesgos:
  - Identifica posibles riesgos a cerca de los requerimientos y las acciones a tomar en cada escenario.
  - Acta de aprobación de la visión.

Planeación (cronograma del proyecto aprobado)

En esta fase la mayor parte de la planeación para el proyecto se termina. El equipo prepara las especificaciones funcionales, realiza el proceso de diseño de la solución, y prepara los planes de trabajo, estimaciones de costos y cronogramas de los diferentes entregables del proyecto

Objetivo:

Obtener un cronograma de trabajo que cumpla con lo especificado en la fase de Visión dentro del presupuesto, tiempo y recursos acordados. Este cronograma debe identificar puntos de control específicos que permitan generar entregas funcionales y cortas en el tiempo.

Entregables:

- Documento de cronograma.
- Acta de aprobación de cronograma.

Desarrollo (alcance completo)

Durante esta fase el equipo realiza la mayor parte de la construcción de los componentes (tanto documentación como código), sin embargo, se puede realizar algún trabajo de desarrollo durante la etapa de estabilización en respuesta a los resultados de las pruebas. La infraestructura también es desarrollada durante esta fase.

Objetivo:

Obtener iterativamente de la mano de la fase de Planeación y de la de Estabilización versiones del producto entregables y medibles que permitan de cara al cliente probar características nuevas sucesivamente. Esto incluye los ajustes de cronograma necesarios.

Entregables:

- Fuentes y ejecutables.
- Documentos manuales técnicos, de usuarios y de instalación si es necesario.
- Acta de finalización de desarrollo.
-

Estabilización (versión aprobada)

En esta fase se conducen pruebas sobre la solución, las pruebas de esta etapa enfatizan el uso y operación bajo condiciones realistas. El equipo se enfoca en priorizar y resolver errores y preparar la solución para el lanzamiento.

Objetivo:

Obtener una versión final del producto probada, ajustada y aprobada en su totalidad.

Entregables:

- Documento registro de pruebas.
- Acta de aprobación de versión aprobada.

Instalación (entrega)

Durante esta fase el equipo implanta la tecnología base y los componentes relacionados, estabiliza la instalación y obtiene la aprobación final del cliente

Objetivo:

Entregar (instalar) al cliente el producto finalizado en su totalidad. Como garantía se han superado con éxito las etapas anteriores.

Entregables:

- Conjunto de archivos (ejecutables directorios, archivos, bases de datos, instaladores, scripts, manuales, licencias, etc.) propios del producto que permitan su instalación y correcto funcionamiento.
- Acta de entrega y finalización del proyecto.

Soporte (entrega ajustada)

Objetivo:

Brindar soporte y garantía al producto durante el tiempo estipulado en el contrato; registrando los reportes de soporte y mantenimiento recibidos, así como los ajustes y versiones ajustadas obtenidas. Esto sólo será valido para ajustes que estén dentro de lo descrito en los documentos de la fase de Visión.

Entregables:

- Documento de registro de reportes de soporte y mantenimiento y ajustes hechos.

Ventajas de MSF

- Es un proceso que se basa en la colaboración entre todos los que realizan el proyecto.
- Analiza riesgos y brinda plantillas que nos ayudan a la hora de la documentación.
- Sirve para grandes y pequeños proyectos.

Desventajas de MSF

- La documentación esta muy grande por lo que necesita una gran paciencia si se trabaja con este proceso.
- El campo de trabajo solo se limita a usar herramientas de este mismo proveedor.
- Es un trabajo bastante largo, ya que para cada fase se debe documentar profundamente todo lo que haga, pero no deja de ser un modelo que tiene buenos resultados.

2.3.3 Proceso Unificado Básico (BUP).

Esta metodología es una versión aerodinámica creada por International Business Machines (IBM) del Proceso Unificado Racional (RUP) perfeccionada para proyectos pequeños que estén conformados por equipos de 3 a 6 personas y con un tiempo de duración de 3 a 6 meses de desarrollo. BUP conserva las características esenciales de RUP que incluye centrado en la arquitectura, iterativo e incremental y dirigido por casos de uso. La mayoría de las partes optativas de RUP se han excluido y muchos elementos se han unido. El resultado es un proceso mucho más simple que RUP.

Está organizada en 2 diversas (pero se correlaciona) dimensiones: métodos y procesos. La dimensión del método es donde se definen los elementos de este (roles, tareas, artefactos, y guía) sin importar cómo se aplican en un ciclo de vida del proyecto.

Disciplinas que plantea BUP

El contenido del método de BUP se centra en las disciplinas siguientes: requisitos, arquitectura, desarrollo, prueba, gerencia de proyecto y gerencia del cambio. El contenido de estas disciplinas se organiza en paquetes, permitiendo la selección solamente del contenido deseado al crear una configuración para publicar.

El contenido del análisis y del diseño fue absorbido por otras disciplinas como arquitectura y desarrollo, esto ocurre porque el arquitecto hace un análisis de alto nivel cuando identifica las abstracciones dominantes, un soporte de alto nivel, la reutilización de soluciones existentes y patrones. El diseño de alto nivel también es hecho por el arquitecto cuando se identifican los componentes importantes y sus interfaces.

Por otra parte, el desarrollador realiza un análisis y diseño de bajo nivel, identificando clases y partes internas de los componentes. La disciplina de desarrollo concentran tareas que el desarrollador realiza para lograr la puesta en práctica del diseño, que es la unidad probada e integrada en la base de código.

El diseño por adelantado no se acentúa en BUP. Este puede ocurrir antes o al mismo tiempo que el desarrollador escribe código y realiza pruebas de unidad. El énfasis está más en el pensamiento del diseño que capturándolo en modelos visuales.

Roles definidos por BUP

BUP en el transcurso del proceso de desarrollo define los siguientes roles:

Analista: responsable de recolectar requisitos y documentarlos según lo necesitado. En proyectos ágiles pequeños, un representante del cliente puede desempeñar este papel.

Arquitecto: responsable de la arquitectura del software, que incluye las decisiones técnicas dominantes que obligan el diseño total y la puesta en práctica para el proyecto.

Desarrollador: crea una solución (o parte de ella) haciendo diseño, la puesta en práctica, pruebas de unidad y la integración de componentes.

Probador: responsable de probar el sistema desde una perspectiva más grande que el desarrollador, debe cerciorarse que el sistema trabaje según lo definido y sea aceptado por el cliente.

Líder del proyecto: organiza y administra el proyecto, coordina interacciones con los clientes, y mantiene al equipo de proyecto enfocado.

Cualquier rol: representa cualquier persona en el equipo que puede realizar tareas como la presentación y la realización de cambios, participando en reuniones, revisando, etc.

Conceptos importantes que define BUP

A continuación se detallan algunos conceptos importantes desde el punto de vista de esta metodología:

Tareas

Solamente las tareas de bajo nivel son consideradas en BUP, aquellas que típicamente son necesarias para proyectos pequeños. Algunas tareas de RUP fueron transformadas en pautas, siendo referidas por tareas más simples.

Algunas de las tareas de RUP fueron transformadas en pasos e incluidas dentro de otra tarea importante que era realizada por el mismo rol en igual tiempo. Un proyecto puede decidir si ese paso se realiza o no, dependiendo de cuál es necesario.

Artefactos

El bajo nivel es obtenido no solamente por el número reducido de los artefactos (comparados a RUP), sino también porque solamente 6 de estos artefactos tienen realmente plantillas. Los artefactos restantes tienen pautas para explicarles cómo los representan informalmente. En general, estas pautas recomiendan la captura de la información existente en un artefacto, una hoja de balance, una base de datos, una tabla, un correo electrónico, etc. Esto permite que los proyectos seleccionen el nivel apropiado y más bajo de formalidad requerido para los artefactos.

Procesos

El contenido reutilizable del método se crea aparte de su uso en los procesos. Este contenido proporciona las explicaciones paso a paso, describiendo cómo las metas específicas alcanzadas del desarrollo son independientes de la colocación de estos pasos dentro de un ciclo de vida de desarrollo.

Los procesos toman los elementos del método y los relacionan dentro de las secuencias que se modifican para requisitos que son particulares de los tipos específicos de proyectos. Los elementos del método se organizan dentro de pedazos reutilizables de patrones llamados proceso de capacidad, proporcionando un acercamiento constante del desarrollo a los problemas comunes. Los patrones pueden ser pequeños y enfocados en áreas particulares como: gerencia de la iteración, iniciación del proyecto, definición de la arquitectura, etc. Éstos se consideran los bloques básicos para crear patrones o procesos más grandes de entrega.

Una entrega de proceso describe un ciclo de vida completo del proyecto y se utiliza como referencia para proyectos similares.

Fases que define BUP.

Inicio

Es la fase más pequeña del proyecto. Si la fase de inicio fuera larga entonces se clasificaría como una fase inicial excesiva, que es contraria al espíritu del proceso unificado.

Objetivos en la fase de Inicio:

- Entender lo que se quiere construir.
- Identificar la funcionalidad dominante del sistema.
- Determinar por lo menos una solución posible.
- Estimar el coste, el horario y los riesgos asociados al proyecto.

Elaboración

En esta fase el equipo de proyecto captura gran parte de los requisitos del sistema. Tiene como meta principal establecer y validar la arquitectura del sistema. Los procesos comunes emprendidos en esta fase incluyen la creación de los diagramas de caso de uso, los diagramas conceptuales y los diagramas del paquete.

El objetivo final de esta fase es entregar un plan para la fase de la construcción. A este punto el plan debe ser exacto y creíble, ya que debe estar basado en la experiencia de la fase de elaboración y los factores de riesgo significativos se deben haber tratado durante esta fase.

Objetivos en la fase de Elaboración:

- Obtener una comprensión más detallada de los requisitos.
- Diseñar, implementar y validar la línea base de la arquitectura.
- Mitigar los riesgos esenciales, producir un calendario preciso y estimar el costo.

Construcción

Esta fase es la más grande del proyecto. Los diagramas comunes de UML usados durante esta fase incluyen diagramas de descripción de la actividad, de la secuencia, de colaboración, del estado (transición) y de la interacción.

Objetivos en la fase de Construcción:

- Desarrollar iterativamente un producto completo, que esté listo para la transición a la comunidad de usuarios.
- Minimizar los costos de desarrollo y lograr un cierto grado de paralelismo.

Transición

Constituye la fase final del proyecto. Esta fase incluye conversiones del sistema y el entrenamiento de usuarios.

Objetivos en la fase de Transición:

- Lograr la concurrencia si el despliegue se ha completado.
- Realizar prueba Beta es para validar si se cumplieron las expectativas del usuario.

Comparación entre las metodologías Tradicionales y Ágiles.

Las metodologías se dividen en dos tipos, en la tabla que se muestra a continuación se desglosan una serie de características que servirán para establecer una breve comparación entre estas:

Metodologías Tradicionales	Metodologías Ágiles
Plantea un proceso disciplinado sobre el desarrollo del software y de esta manera sea más eficiente.	Encuentra un punto de equilibro entre no-documentación y mucha-documentación.
Detallan procesos con énfasis en la planeación.	Se obvia toda esta parte de la documentación y se enfoca en dar resultados.
Se resisten a los cambios.	Brinda cambios y resultados continuos.
Se espera que este tipo de metodología sea predecible.	Son adaptables, no predictivos.
Es un proceso mucho más controlado, que presenta numerosas políticas/normas.	Se orientan a los individuos, no a los procesos.
Son impuestas externamente, es decir, no son impuestas por el equipo de desarrollo.	Se conforman por grupos pequeños, por lo general menos de 10 integrantes y trabajan en el mismo sitio.
La arquitectura del software es esencial y se expresa mediante modelos.	Especialmente preparados para cambios durante el proyecto.
Existe un contrato prefijado.	Son impuestas internamente por el equipo de desarrollo.
Están basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.	Proceso menos controlado, con pocos principios.
Trabaja con grandes grupos de desarrollo.	No existe contrato tradicional o al menos es bastante flexible.
El cliente interactúa con el equipo de desarrollo mediante reuniones.	El cliente es parte del equipo de desarrollo.
Es el tipo de metodología que más roles define.	Definen pocos roles.
Utilizan muchos más artefactos.	Utilizan pocos artefactos.

CAPÍTULO 3. PROPUESTA DEL PROCESO DE DESARROLLO DEL SMQ

En este Capítulo se define el proceso de desarrollo del software para un Simulador Quirúrgico, el cual fue elaborado partiendo de las etapas y actividades que habían definidas hasta el momento para el proyecto SMQ.

Tomando como antecedente lo planteado en el capítulo anterior respecto a las metodologías de desarrollo tanto las tradicionales como las ágiles, se identificaron los elementos de estas metodologías que pueden ser aplicables al proceso de desarrollo del SMQ y que tributan a su perfección.

3.1 Aportes al Proceso definido.

A continuación se describen los elementos fundamentales del proceso que guían el proceso definido.

- **Definir el guión técnico del producto antes de comenzar las etapas de desarrollo.**

Los guiones más usados en los Sistemas de Realidad Virtual son los de videojuegos, este guión se divide en dos partes, una primera, llamada guión de contenido que es donde se detalla la trama del juego, o sea, la historia que le da soporte, aquí se especifican los personajes, los antecedentes de dicha historia, el flujo de misiones, en fin, todos los elementos narrativos. Y la otra parte es el guión técnico, también conocido como documento de diseño, que es donde se establece la vinculación entre la trama y el software, aquí se describen a fondo las interacciones del usuario y el sistema, los controles, el entorno y los elementos de este, entre otros que especifican las ideas que más tarde van a implementar los desarrolladores (programadores, diseñadores 3D, etc.), por eso es importante no dejar escapar ningún detalle.

Después de haber analizado los elementos que conforman un guión de videojuegos, se valoró la posibilidad de recomendar el uso de un guión técnico para el proceso de desarrollo de un SMQ, atendiendo a que durante el proceso actual que se lleva a cabo en el KHEIPROS se han presentado inconvenientes, sobre todo a la hora de hacer el diseño de los casos de pruebas, por no tener detallado a grandes rasgos cada funcionalidad. Claro que, este guión técnico tiene diferencias marcadas con el de videojuegos, pero en general persigue el mismo objetivo, de abordar cada idea a desarrollar en el software, lo cual sin duda ayuda a que los desarrolladores puedan trabajar guiándose

por lo escrito en el guión sin tener que estar pidiendo detalles de su temática a los jefes de equipo y el líder del proyecto.

El rol de Guionista en este caso merece una definición, pues difiere del guionista de juegos que tradicionalmente se conoce. Aquí no se está hablando de una persona con aptitudes para escribir e inventar historias interesantes, sino que, este guionista conoce a fondo los requerimientos del sistema y que también posee una amplia imaginación dada por la experticia en la temática, que le permiten visualizar en su mente previamente las interfaces, los entornos, los elementos de la escena, el flujo de mensajes, todas las posibles interacciones que pueden existir con el usuario y las respuestas que el sistema debe darle. De aquí resulta también que la elaboración de dicho guión debe ser un trabajo conjunto entre el guionista, diseñadores 3D, programadores, analista, líder del proyecto, o sea, que deben participar todas las personas que se vean implicadas en lo que se va a plasmar en el guión.

La idea es que las cuatro etapas de desarrollo del producto con las que hasta ahora cuenta el SMQ tengan incluido un guión técnico, es decir, habrá un guión para el Simulador de habilidades básicas, uno para el de habilidades avanzadas, uno para obtener la primera versión del simulador a partir de la generación de órganos virtuales y uno para obtener un Simulador con órganos adquiridos a partir de scanner (TAC, RMI) y retroalimentación al tacto. En el caso del guión que se define para la última etapa de desarrollo no puede contar con un alto nivel de detalle, o sea, todo tiene que quedar expuesto de manera muy general, algo genérico que recoja los aspectos comunes para todos los posibles escenarios, pues el producto a desarrollar en esta etapa es un simulador capaz de soportar la planificación previa de la cirugía de determinada patología diagnosticada en un paciente real, por tanto, aquí en el flujo de actividades es específico para cada caso. Claro que, estas patologías están delimitadas en un área o tema específico de la cirugía para la que fue confeccionado el Simulador, pero aún así son escenarios muy específicos.

En esta actividad se obtiene una primera versión de la plantilla para elaborar el guión de un Simulador, que recoge las ideas que se han determinado como básicas a plasmar en el guión (Ver anexo 4).

- **Actividades de aseguramiento de la calidad en cada etapa.**

Partiendo del objeto social de un Simulador Quirúrgico, que es el de adiestrar cirujanos y residentes en la especialidad, para que más tarde estos puedan ejercer en operaciones con pacientes reales las

habilidades que adquirieron con el sistema, salta a la vista la necesidad de que cada componente del simulador ya sea de hardware o de software brinde al usuario funcionalidades con un alto nivel de precisión y calidad. Por tanto, en el proceso definido en esta investigación se propone que constantemente se realicen actividades de aseguramiento y control de la calidad del producto y el proceso, para este último es importante la definición y aplicación de listas de chequeo y revisiones que permitan validar el correcto desarrollo del flujo de actividades y esto a su vez posibilita retroalimentar a los desarrolladores y así perfeccionar el proceso. Para garantizar la calidad del producto se recomienda que se hagan las pruebas requeridas a los resultados que se van obteniendo. Todas estas actividades de aseguramiento y control de la calidad se deben hacer según lo planteado en la tesis *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"*. Sólo señalar que a partir de la segunda etapa de desarrollo se hace imprescindible que el especialista participe en la etapa de pruebas del producto, ya que él es quien puede brindar el criterio de experto para comprobar la calidad de la solución, es por esto que en la etapa inicial del proceso donde se pactan las tareas con el cliente se debe especificar por escrito los acuerdos para dar cumplimiento a esto.

- **Identificar necesidades de capacitación en cada etapa.**

Se recomienda que al hacer el Levantamiento de los Riesgos en cada etapa, se tengan en cuenta las necesidades de capacitación que tiene el proyecto para enfrentar las próximas actividades de desarrollo, de manera que con tiempo se puedan planificar cursos, talleres, encuentros, que sirvan para capacitar los miembros del equipo y garantizar la fluidez del desarrollo, en tiempo y forma.

- **Insertar el rol de Responsable de vigilancia tecnológica**

Este rol es el encargado de velar por el estado de la línea temática trabajada en el proyecto, sus tendencias en el mundo, las tecnologías usadas y las nuevas que van surgiendo, las posibles opciones de mercado, las figuras que constituyen fuentes potenciales de colaboración. En fin, los aspectos del entorno que se pueden aprovechar para perfeccionar y actualizar el producto y además para incorporarlo al mercado.

3.1.1 Elementos de las Metodologías que se pueden integrar al proceso propuesto.

El proyecto SMQ está estructurado según plantean las características de la metodología RUP, la misma está diseñada para aplicarse a proyectos robustos, extensos, pesados, guiado por casos de uso, con iteraciones e incrementos, centrados en la arquitectura, abundante documentación, etc. Luego de estudiar y analizar en el capítulo anterior varias metodologías, se puede concluir que es RUP la ideal para continuar aplicándose en el proceso de desarrollo, sin dejar de señalar que pueden incorporarse elementos de otras metodologías a la hora de implementar determinadas actividades y de esta manera asegurar la eficiencia del producto.

- **Gestión del conocimiento.**

XP dentro de sus principales características plantea que cada uno de sus miembros debe tener conocimiento mínimo de cada tarea y/o actividad que se está realizando en el proyecto para que en caso que falte alguno de sus integrantes, la función que este estaba realizando no se detenga; esto se ve reflejado como una dificultad en el proceso de desarrollo actual del SMQ debido a que gran parte del equipo que lo conforma llega al último año de la carrera y tiene que retirarse de la Universidad y por tanto del proyecto, llevándose consigo todos los conocimientos que adquirió en determinada línea de desarrollo. Por lo cual se hace necesario gestionar ese conocimiento y socializarlo a otros miembros del equipo. Para esto se propone hacer talleres y exposiciones donde el experto exponga las técnicas que se emplean para desarrollar la solución y los avances que se van teniendo. También se puede vincular a esto los estudiantes que estén en proceso de formación en el proyecto y orientarle tareas sobre estos temas que están en desarrollo y que dichas tareas sean supervisadas por estos expertos.

- **Trabajar en parejas.**

MSF en su Modelo de Roles defiende la idea de que el proceso esté enfocado a estructurar a los integrantes del equipo de trabajo y sus actividades con el objetivo de alcanzar el éxito del proyecto, para esto apoya el trabajo en parejas ya que de esta manera pueden surgir muchas más ideas que aporten a la solución de las actividades a realizar, vale aclarar que esta es una de las características más distintivas de esta metodología.

Por otra parte está XP que en una de sus características plantea que el trabajo debe ser realizado por parejas, es decir, uno de los integrantes define como se va a desarrollar la actividad y el otro se encarga de pensar las posibles vías y estrategias que permiten desarrollar la actividad de una forma más sencilla y que esta a su vez sea robusta.

Este trabajo en parejas trae consigo muchos beneficios pues en caso de existir alguna deficiencia es más fácil detectarla ya que existe una persona revisando el código continuamente, por lo que se tiene un menor número de errores en la actividad realizada; cada miembro tiene ideas diferentes por lo que al complementar sus ideas el trabajo quedaría más completo y mejor.

Esta característica no se trataría de igual forma en el proyecto SMQ, o sea, se trabajaría en parejas pero en horarios diferentes debido a la escasez de máquinas que existe, lo que trae como consecuencia, que un dúo no pueda utilizar dos máquinas al mismo tiempo.

Las mejoras antes expuestas reflejan que en algunos casos es una buena práctica aplicar características de otras metodologías al proceso en cuestión, con el 0fin de mejorar el proceso de desarrollo del software en el SMQ.

3.2 Proceso de desarrollo del SMQ.

En este epígrafe se define el Proceso de Desarrollo para el proyecto SMQ, que fue el objetivo de este trabajo. Este proceso está basado en las etapas y actividades que habían definidas hasta el momento en el proyecto SMQ, partiendo de esto la idea fue perfeccionar lo que había definido y elaborar un proceso que abarque todas las etapas por las que pasa el desarrollo del producto y que el mismo sirva de guía para el equipo de desarrollo, pues en él se plasma paso a paso cada una de las actividades necesarias para obtener un producto final.

A partir del estudio realizado en la fundamentación teórica se fueron definiendo las características del proceso, lo primero fue identificar que el modelo al que se ajusta este proceso es el Incremental el cual pertenece a la categoría de los Modelos Evolutivos, los que surgen debido a la necesidad de adaptar modelos de procesos a productos que evolucionen con el tiempo, además son iterativos y se caracterizan por permitir desarrollar versiones cada vez más completas del software. Por su parte la norma ISO/IEC 12207 permitió validar la efectividad de las actividades que estaban definidas

previamente en el proyecto SMQ. También se tomó cómo guía lo plateado en el PMBook para la Gestión de proyectos que fue muy útil para definir la parte inicial del proceso. Y por último las metodologías estudiadas brindaron algunas prácticas, que como se mencionaba en el epígrafe anterior, se proponen implementar en el proceso para complementarlo y hacer más eficiente el desarrollo de las actividades.

Es importante aclarar que la implementación de cada una de estas actividades debe estar sujeta a lo que plantea la metodología RUP para cada caso específico, pues es la metodología de desarrollo que se utiliza en el proyecto SMQ, excepto en el caso de las actividades que se vean afectadas por lo elementos del proceso que se mencionaron en el epígrafe anterior que pertenecen a otras metodologías, en este caso se tratará de lograr el equilibrio entre lo que necesariamente se deba hacer por RUP, pues esta es la metodología base y las otras prácticas propuestas.

3.2.1 Fases del proceso de desarrollo del SMQ.

En este epígrafe quedarán definidas las fases que conforman el proceso de desarrollo del SMQ y el flujo de actividades para cada etapa. Debido a lo complejo y extenso que llega a ser el proceso de desarrollo definido para el SMQ, es necesario apoyarse en diferentes diagramas donde queden reflejadas de forma más resumida las actividades, facilitando la comprensión del mismo. A continuación se muestran 6 diagramas, el primero muestra cómo está conformada la fase de concepción del proyecto, el segundo comienza con la fase de desarrollo, la misma está organizada en cuatro etapas, donde cada una de ellas está representada por un diagrama, según las actividades que estén definidas, y el último refleja la fase final del proyecto.

Roles definidos en el proceso de desarrollo

Para realizar las actividades que se definen en el proceso de desarrollo del software se utilizan los roles que se describen en el epígrafe 1.6.1 además de insertarse el rol de Guionista y el Responsable de vigilancia tecnológica.

FASE DE CONCEPTUALIZACIÓN

La fase de conceptualización es donde se concibe y delimita el proyecto, describiendo su alcance y dominio. Esta fase muestra los primeros pasos que se deben realizar en el proceso de desarrollo que permiten hacer la planificación de un proyecto.

Es importante señalar que pueden existir dos escenarios a la hora de comenzar esta etapa, uno es tener un proyecto que logre desarrollar un producto de este tipo pero no cuente con un cliente específico, en este caso se está en presencia de un proyecto a riesgo, por lo que necesariamente hay que realizar un profundo estudio de mercado de la temática a desarrollar e identificar las competencias necesarias que debe tener el producto y las líneas de desarrollo más viables en las que se debe adentrar el proyecto, también para identificar posibles nichos de mercado, en caso de que el producto se realice con fines comercializables. Y el otro escenario es el de un proyecto con cliente definido y que por tanto sea este quien dicte las necesidades a satisfacer por el producto.

En el caso que una institución esté interesada en adquirir el Simulador, lo primero que se debe hacer es la actividad que se describe a continuación.

Actividad: Pactar tareas con el cliente.
Responsable: Líder de Proyecto.
Participante: Líder de Proyecto.
Entradas:

- Cuestionario de entrevista.

Salidas:

- Descripción del producto a desarrollar.
- Compromisos de ambas partes para enfrentar las etapas de desarrollo del software.

Descripción:
En esta actividad el cliente plantea al Líder de Proyecto las necesidades que presenta y por las que solicita el producto, a medida que se desarrolla la entrevista ambas partes llegan a acuerdos sobre el tiempo de desarrollo que tendrá el producto. Se debe definir una lista de compromisos inicial que recoja las principales responsabilidades de ambas partes durante el proceso de desarrollo.

Actividad: Realizar un estudio del estado del arte.
Responsable: Responsable de vigilancia tecnológica.
Participante: Responsable de vigilancia tecnológica.

Entradas:

- Misión del proyecto.

Salidas:

- Estado del arte de la temática.
- Líneas de desarrollo hacia donde debe moverse el proyecto.
- Caracterización de las principales organizaciones y empresas líderes en la temática e ideas iniciales para establecer posibles relaciones con las mismas.
- Estimación inicial de los costos de desarrollo.

Descripción:

Esta actividad garantiza que se realice un estudio constante sobre la tendencia actual del desarrollo de los Simuladores Quirúrgicos, permitiendo con esto hacer estimaciones sobre cuáles tienen más aceptación en el mercado.

Actividad: Confeccionar el Plan del proyecto.

Responsable: Líder de Proyecto.

Participantes: Líder de Proyecto.

Entradas:

- Metodología de desarrollo.

Salidas:

- Plan del proyecto.
- Glosario de términos.

Descripción:

Esta actividad proporciona un marco de trabajo que le permite al responsable hacer un proceso de descubrimiento de la información que lleve a estimaciones razonables de recursos, costes y planificación. Estas estimaciones se hacen dentro de un marco de tiempo limitado. Este plan está conformado por el ámbito del software, que describe el control y los datos a procesar, la función, el rendimiento, las restricciones, las interfaces y la fiabilidad del mismo; la viabilidad, que decide si se podrá construir el software de acuerdo a ese ámbito, es decir, no siempre lo que se piensa respecto al rendimiento es factible; los recursos principales a considerar antes de empezar el proyecto son los humanos y los de entorno, aunque existen otros.

Debido a la complejidad que presenta un proyecto de este tipo (SMQ), es necesario trabajar en base a una exquisita calidad del producto, pues cualquier deficiencia podría ser catastrófica, por esta razón es necesario hacer un **Plan de aseguramiento de la calidad** en cada etapa de desarrollo para trabajar con éxito desde que se inicia el proyecto. La calidad del producto se puede expresar como eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad. El Proyecto SMQ cuenta con un trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"* para dar cumplimiento a esta actividad.

El **Plan de gestión de configuración** garantiza la integridad del producto a lo largo del desarrollo del software. Es muy importante que se realice esta actividad ya que permite establecer las políticas para el control de cambios, define cuáles son los roles que se relacionan con el proceso de Gestión de Configuración del Software e identifica los elementos del mismo, permite escoger e instalar las herramientas para el control de los cambios y las versiones y posibilita crear las líneas bases. Esta actividad se desarrolla en el trabajo de diploma "*Gestión de Configuración para el proyecto Simulador Quirúrgico*".

Actividad: Definir los riesgos en las etapas de desarrollo.
Responsable: Planificador de la calidad.
Participantes: Planificador de la calidad, Líder de Proyecto.
Entradas:

- Plan del proyecto.

Salidas:

- Lista de riesgos.
- Líneas de investigación.

Descripción:

Esta actividad permite identificar las fuentes de riesgos antes de que comiencen a amenazar el éxito o la finalización exitosa del desarrollo del software ya que, un riesgo es la posibilidad de que un evento adverso o algún contratiempo pueda ocurrir produciendo pérdidas. También es importante destacar que aquí deben definirse las necesidades de capacitación en cuanto a habilidades y conocimientos de los recursos humanos. Identificar qué temas de la solución se hace necesario investigar debido a que no se cuenta con experticia en él, de esto salen las líneas de investigación que serán cubiertas más tardes por los investigadores del proyecto. Para dar cumplimiento a lo planteado en el epígrafe 1.6.1 se

propone que esta actividad se retome al inicio de cada etapa de desarrollo con el objetivo de chequear si hay nuevos riesgos y cuáles de los identificados al inicio fueron mitigados.

Actividad: Análisis y mitigación de los riesgos.
Responsable: Planificador de la calidad.
Participantes: Planificador de la calidad, Líder de Proyecto.
Entradas:

- Lista de riesgos.

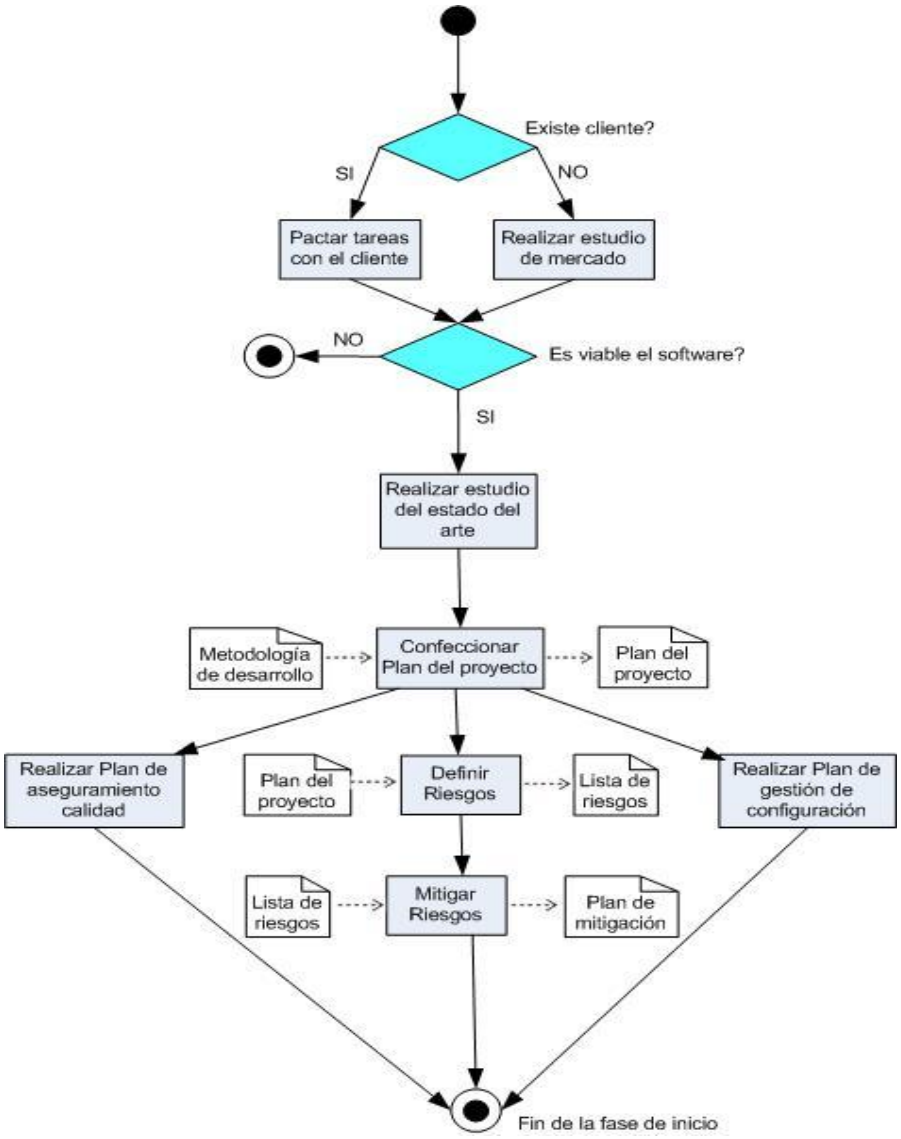
Salidas:

- Plan de mitigación de riesgos.
- Plan de capacitación.

Descripción:

Esta actividad está guiada a identificar, estudiar y eliminar los riesgos que puedan afectar el desarrollo exitoso del software. De acuerdo al avance en el desarrollo del proyecto en que se encuentra el riesgo existen cinco niveles: control de crisis, es controlar los riesgos cuando ya se han convertido en un problema; arreglar cada error, es reaccionar rápidamente ante la presencia de cualquier amenaza; mitigación de riesgos, se planifica con antelación el tiempo que se necesitaría para cubrir un riesgo ocurrido; prevención de riesgos, se identifican los posibles riesgos antes que estos se conviertan en problemas y eliminación de las causas principales que se encarga de identificar y eliminar las posibles causas de un riesgo. Debido a que el riesgo es una posibilidad futura, un buen análisis puede determinar la ocurrencia o no ocurrencia de los mismos. Esta actividad al igual que la anterior se debe retomar en cada etapa.

A continuación se muestran las actividades que se desarrollan en esta fase:



FASE DE DESARROLLO

En esta fase se desarrollan actividades de software y se definen las cuatro etapas que hasta el momento tiene el SMQ.

Actividades correspondientes a la PRIMERA ETAPA de desarrollo del software:

La captura, análisis y especificación de requisitos es un eslabón fundamental en la realización de un software, es por esa razón que la primera actividad que se realiza al comenzar la fase de desarrollo es el **Levantamiento de requisitos,** de este resultado depende en gran medida el logro de los objetivos finales. Esta actividad está desarrollada con gran nivel de detalle en el trabajo de diploma *"Definición y aplicación de las técnicas adecuadas para el desarrollo de Requisitos en el Simulador Quirúrgico",* donde se utiliza una nueva estrategia para realizar esta actividad, los resultados que se obtienen se guardan en el documento de Especificación de Requisitos de Software (ERS) el cual está archivado en el *repositorio* del SMQ. Es muy importante aclarar, que esta actividad se realiza al inicio de cada etapa de desarrollo.

Actividad: Emprender las investigaciones de las actividades a realizar en las etapas de desarrollo.
Responsable: Investigador.
Participante: Investigador, Líder de Proyecto.
Entradas:

- Documento ERS.

Salidas:

- Documentación.

Descripción:
Estas investigaciones son precisamente para conocer qué debe hacerse en cada actividad y cómo hacerlo. Si durante la investigación de los algoritmos, el investigador detecta que alguno no es viable, si no se ha comenzado a trabajar en él no se realiza, de lo contrario se detiene inmediatamente su ejecución y en caso que sea indispensable utilizarlo y no se pueda encontrar otra vía de solución para el mismo se detiene el desarrollo del proyecto. La información que finalmente se selecciona es almacenada en el repositorio. Esta actividad se realiza al inicio de cada etapa de desarrollo.

Existe la posibilidad que para investigar los temas de la etapa de desarrollo en cuestión, no sea necesario afectar a todos los investigadores del proyecto, por lo que se puede dividir el equipo de investigadores para que un grupo realice paralelamente las investigaciones correspondientes a la próxima etapa de desarrollo. En caso contrario, al finalizar las investigaciones de la etapa en cuestión se comienza a investigar en los temas de la próxima etapa de desarrollo y así sucesivamente con las siguientes etapas.

Actividad: Revisión Técnica Formal.
Responsable: Planificador de la calidad.
Participante: Planificador de la calidad, Líder de Proyecto, Documentador, Investigador.
Entradas:

- Glosario de términos.
- Documentación.
- Lista de chequeo.
- Documento ERS.

Salidas:

- Informe con los resultados obtenidos en la revisión.

Descripción:
Esta actividad permite realizar un chequeo a la actividad anterior para detectar los posibles errores que puedan existir teniendo en cuenta la lista de chequeo definida en el trabajo de diploma “Propuesta de una estrategia de aseguramiento de la calidad para el SMQ”, esta lista permite medir la calidad del proceso.

Nota importante: En la primera etapa de desarrollo del proyecto se trabaja para lograr un Simulador de habilidades básicas, a partir de modelos de imitación de procesos. Este Simulador estará conformado principalmente por tres ejercicios (Focalización de la Cámara, Coordinación mano - ojo y Selección y Transferencia) los cuales están enfocados a la puesta en práctica de algunas de las habilidades básicas que deben alcanzar los cirujanos durante su adiestramiento. A continuación tienen lugar las actividades correspondientes a esta etapa:

Actividad: Realizar el guión.
Responsable: Guionista.

Participante: Guionista, Líder de Proyecto, Programador, Diseñador 3D, Analista principal, Administrador y configuración.

Entradas:

- Documento ERS.

Salidas:

- Guión técnico del ejercicio.

Descripción:

El guionista se reúne con los participantes y entre todos definen el guión, dejando claro la descripción detallada de las funcionalidades, los entornos de las escenas, los elementos de la interfaz de usuario, los controles para interactuar con el software, el flujo de sucesos del ejercicio, etc., es decir, lo que el equipo tendrá que hacer para crear el simulador. En fin, los elementos del guión permitirán guiar el proceso de desarrollo del ejercicio a partir de los requisitos capturados.

Actividad: Realizar el Análisis y el Diseño del software.

Responsable: Analista principal.

Participante: Analista principal, Líder de Proyecto, Analista de Software, Arquitecto de Software, Diseñador 3D, Diseñador de Interfaz de usuario.

Entradas:

- Descripción de Casos de Uso.
- Documentación.
- Documento ERS.

Salidas:

- Propuesta de la Arquitectura.
- Casos de Uso aprobados.
- Diagrama de clases que corresponden al Análisis y al Diseño.
- Modelo de datos.
- Diagrama de interacción.
- Documento final de arquitectura.

Descripción:

Aquí se realiza el proceso de análisis del sistema. Se identifican las clases relacionadas con cada caso de uso, sus atributos y relaciones. Se elabora el documento de realización de los casos de uso y luego

con esta información se elaboran los diagramas del análisis y el diseño. Además se diseña la Base de Datos (BD). En esta etapa se define el Análisis y el Diseño del software.

Actividad: Modelar la interfaz gráfica.
Responsable: Diseñador 3D.
Participante: Diseñador 3D, Líder de Proyecto.
Entradas:

- Guión técnico del ejercicio.

Salidas:

- La interfaz gráfica.

Descripción:
En esta actividad se diseña la interfaz principal del producto, para esto se utiliza la herramienta de diseño 3DMax. Al concluir esta actividad se aplican **Pruebas de caja negra**, con el objetivo de comprobar si se cumple con los requisitos planteados.

Actividad: Modelar el mundo virtual para todos los niveles de complejidad definidos.
Responsable: Diseñador 3D.
Participante: Diseñador 3D, Líder de Proyecto.
Entradas:

- Guión técnico del ejercicio.

Salidas:

- Entorno 3D.

Descripción:
Se modelan los distintos componentes que forman parte de la escena donde se desarrollan los ejercicios, para esto se utiliza la herramienta de diseño 3DMax.

Actividad: Modelar los instrumentos quirúrgicos.
Responsable: Diseñador 3D.
Participante: Diseñador 3D, Líder de Proyecto.
Entradas:

- Guión técnico del ejercicio.

Salidas:

- Instrumentos quirúrgicos modelados en 3D.

Descripción:

Se modelan los diferentes instrumentos quirúrgicos en 3D para simular las herramientas que se utilizan en una operación real, donde se diseñan pinzas para el agarre y selección de objetos para poder realizar los módulos. Al culminar la modelación de los objetos, estos son exportados con extensión 3dx.

Actividad: Diseñar el ejercicio para todos los niveles de complejidad definidos.

Responsable: Programador.

Participante: Programador.

Entradas:

- Guión técnico del ejercicio.
- Diagrama de casos de uso del sistema.

Salidas:

- Artefactos de diseño que propone RUP.

Descripción:

Teniendo en cuenta lo especificado en el guión, se realiza el diseño de los ejercicios que da soporte a todos los niveles de complejidad definidos.

Actividad: Desarrollar algoritmo de colisiones para procesos dinámicos.

Responsable: Programador.

Participante: Programador, Líder de Proyecto, Jefe de Equipo, Arquitectura y tecnologías.

Entradas:

- Documentación.
- Documento ERS.

Salidas:

- Se obtienen colisiones de forma dinámica.

Descripción:

A partir de las investigaciones realizadas se comienza el desarrollo de algoritmos de colisiones para procesos dinámicos donde se obtiene como resultado final un módulo que valida los algoritmos seleccionados. Antes de guardarlo en el repositorio, se aplican **Pruebas de unidad** con el objetivo de detectar los posibles errores que puedan existir. Este tipo de prueba se describe con mayor

profundidad y cómo aplicarla, en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ".* Para la realización de estos ejercicios se utiliza la herramienta STK.

Actividad: Implementar el ejercicio para todos los niveles de complejidad definidos.
Responsable: Programador.
Participante: Programador, Líder de Proyecto, Jefe de Equipo, Arquitectura y tecnologías.
Entradas:

- Documento ERS.
- Artefactos de diseño que propone RUP.
- Los modelos gráficos correspondientes.

Salidas:

- Ejecutable del ejercicio.

Descripción:
A partir de las entradas definidas en esta actividad, se implementa el ejercicio en sus niveles de dificultad definidos según correspondan. En el ejercicio **Focalización de la Cámara** el usuario debe ser capaz de focalizar los cuerpos parpadeantes dentro de una cavidad modelada en 3D en un tiempo determinado creando habilidades con el uso de la cámara. En el ejercicio de **Coordinación mano-ojo** el usuario debe ser capaz de coordinar las pinzas de tal modo que el objeto seleccionado sea el correspondiente. En el ejercicio de **Selección y Transferencia** se crean habilidades con la manipulación de las pinzas ya que es necesario tener precisión al seleccionar los objetos y trasladarlos hacia el lugar donde se le indique al usuario. Al terminar cada ejercicio se desarrolla un módulo para respaldar los resultados obtenidos en la investigación y se aplican **Pruebas de unidad** antes de guardarlo en el repositorio, con el objetivo de detectar los posibles errores que puedan existir. Este tipo de prueba se describe con mayor profundidad y cómo aplicarla, en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ".* Para la realización de estos ejercicios se utiliza la herramienta STK.

Actividad: Diseñar la BD del SMQ.
Responsable: Diseñador principal de base de datos.
Participante: Diseñador principal de base de datos, Arquitectura y tecnologías.
Entradas:

- Documento ERS.

Salidas:

- Diseño de la BD.

Descripción:

En esta actividad se crean las tablas de la BD definida a partir de las variables obtenidas mediante la captura de los requisitos relacionados con las estadísticas a medir en el ejercicio.

Actividad: Implementar el módulo de gestión de la BD.

Responsable: Programador.

Participante: Programador, Diseñador principal de base de datos.

Entradas:

- Documento ERS.
- Diseño de la BD.

Salidas:

- Módulo de gestión de la BD.

Descripción:

Con la realización de esta actividad queda implementado el módulo de gestión de BD según quedó conformado en el diseño. Permite almacenar los diferentes valores que vayan tomando las variables. Al terminar esta implementación se aplican **Pruebas de unidad** antes de guardarlo en el repositorio.

Actividad: Integrar los ejercicios de la etapa.

Responsable: Programador.

Participantes: Programador, Líder de Proyecto, Analista principal.

Entradas:

- Ejecutables de los ejercicios correspondientes a esta etapa.
- Descripción de la arquitectura del software.
- Documento ERS.
- La interfaz gráfica.

Salidas:

- Módulo conformado por los ejercicios de la etapa.

Descripción:

En esta actividad se crea el módulo donde se integran los ejercicios realizados a lo largo de la etapa de desarrollo. A esto le seguiría una actividad donde se aplican **Pruebas de integración** de manera que se pueda comprobar que el módulo conformado por la integración de los ejercicios tuvo éxito, también se aplican **Pruebas que permitan validar las funcionalidades del sistema**, para comprobar que el sistema satisface los requisitos del usuario, tanto los funcionales como los no funcionales; en el trabajo de diploma titulado "*Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"* se describe con mayor profundidad y cómo realizar este tipo de prueba.

Actividad: Hacer una copia de salva del ejercicio.
Responsable: Administrador y configuración.
Participante: Administrador y configuración, Programador.
Entradas:

- Módulo conformado por los ejercicios de la etapa.

Salidas:

- Copia del módulo en el servidor.

Descripción:
Se copia en una carpeta el ejercicio, esta se configura con la dirección del repositorio, para guardar el ejercicio en el servidor se necesita la cuenta de usuario del programador. Siempre que se modifique o se trabaje en el ejercicio se hace necesario actualizar el servidor. Para realizar esta actividad es necesario tener instalada la herramienta *TortoiseSVN* que es utilizada para realizar el control de versiones.

Actividad: Elaborar el manual de usuario.
Responsable: Líder de Proyecto.
Participante: Líder de Proyecto, Programador, Documentador.
Entradas:

- Ejecutables de los ejercicios de esta etapa.
- Módulo de gestión de la BD.

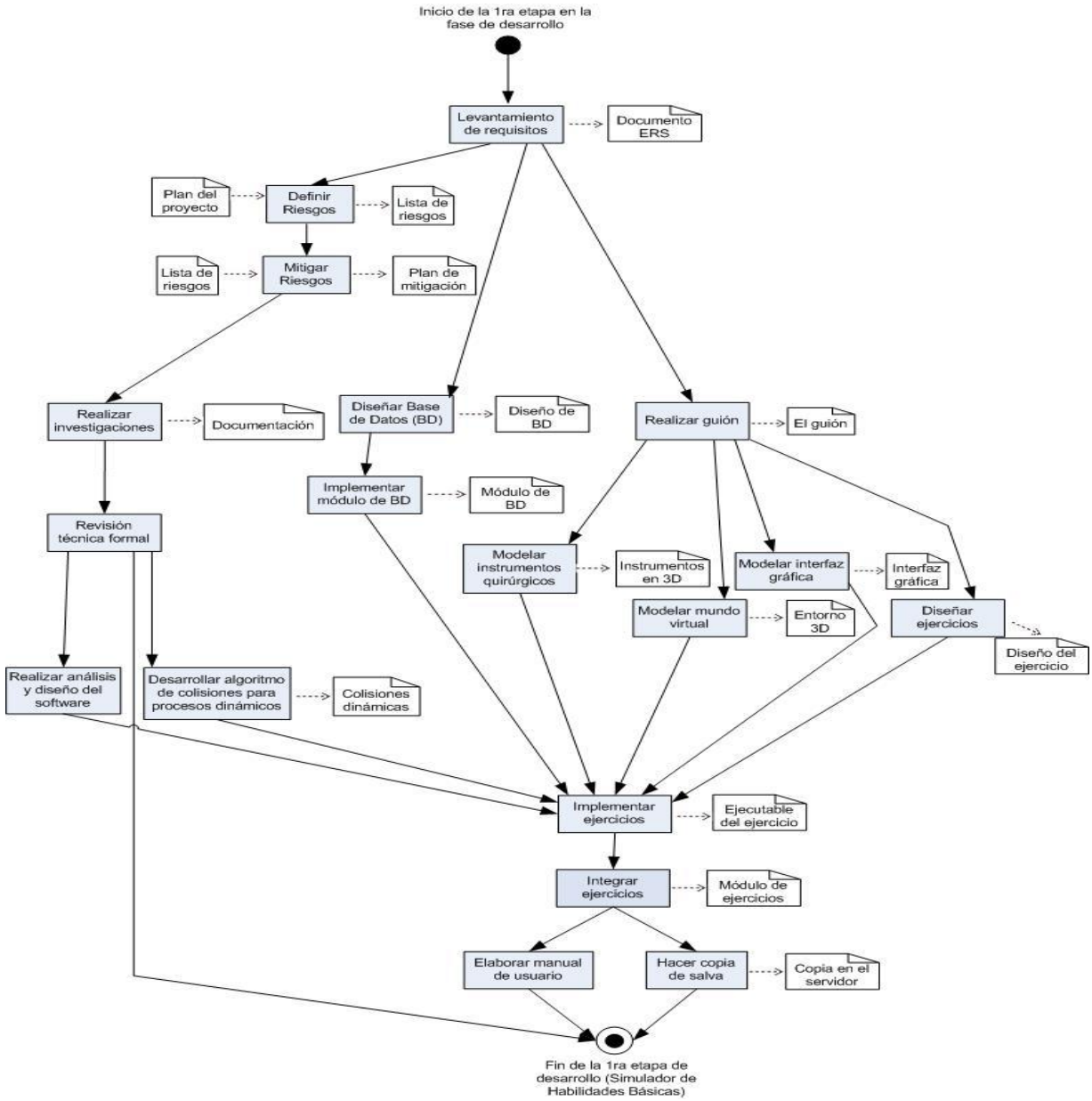
Salidas:

- El manual de usuario.

Descripción:

Se elabora un manual de usuario para esta etapa del proceso de desarrollo del SMQ donde se especifica detalladamente, paso a paso, cómo utilizar el sistema en cada una de las funcionalidades definidas en esta etapa de desarrollo. De esta forma el usuario podrá contar con una guía de apoyo en el momento de la ejecución del sistema.

La figura que a continuación se muestra, refleja las actividades que conforman la primera etapa de desarrollo.



Actividades correspondientes a la SEGUNDA ETAPA de desarrollo del software:

En esta etapa de desarrollo del proyecto se trabaja para lograr un Ejecutable del Simulador de habilidades avanzadas a partir de la imitación de procesos. Este Simulador estará conformado principalmente por tres ejercicios (Corte, Sutura y Grapado), estos tienen como objetivos que los cirujanos desarrollen habilidades avanzadas durante su entrenamiento, por esa razón a partir de esta etapa es imprescindible la presencia de un especialista en cirugía, que realice las pruebas pertinentes para validar en cada módulo el nivel de precisión que deben tener los ejercicios. A continuación tienen lugar las actividades correspondientes a esta etapa:

Actividad: Modelar el mundo virtual para todos los niveles de complejidad definidos.
Nota: Esta se realiza de la misma forma que en la etapa anterior.

Actividad: Modelar los instrumentos quirúrgicos.
Nota: Esta se realiza de la misma forma que en la etapa anterior.

Actividad: Diseñar el ejercicio para todos los niveles de complejidad definidos.
Nota: Esta se realiza de la misma forma que en la etapa anterior.

Actividad: Implementar la generación de fluido.
Responsable: Programador.
Participante: Programador, Jefe de equipo, Arquitectura y tecnologías.
Entradas:

- Documentación.

Salidas:

- Generación de fluido.

Descripción:
A partir de diferentes modelos matemáticos que se pueden utilizar para hacer la simulación de fluido, se seleccionó un método de simulación física de fluidos llamado Hidrodinámica de las partículas suavizadas, con el que se obtuvo un módulo que permitió probar que era factible, aunque se continuará este estudio en busca de modelos más óptimos. Al terminar esta implementación y antes de

guardarlo en el repositorio se aplican **Pruebas de unidad**, con el objetivo de detectar los posibles errores que puedan existir. Este tipo de prueba se describe con mayor profundidad y cómo aplicarla, en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"*.

Actividad: Diseñar el algoritmo de simulación del hilo basado en el modelo matemático.
Responsable: Programador.
Participante: Programador, Arquitectura y tecnologías.
Entradas:

- Documento ERS.
- Documentación.

Salidas:

- Descripción detallada de algoritmo para la simulación del hilo.

Descripción:
Se define con precisión el algoritmo a implementar para lograr la simulación del hilo.

Actividad: Implementar el modelo matemático de simulación del hilo.
Responsable: Programador.
Participante: Programador, Jefe de Equipo, Arquitectura y tecnologías.
Entradas:

- Documento ERS.
- Diseño detallado de la actividad anterior.

Salidas:

- Simulación del hilo.

Descripción:
Se implementarán las clases con sus funcionalidades necesarias para lograr la simulación del hilo. Al terminar esta implementación se aplican **Pruebas de unidad** antes de guardarlo en el repositorio, con el objetivo de detectar los posibles errores que puedan existir. Este tipo de prueba se describe con mayor profundidad y cómo aplicarla, en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"*.

Actividad: Iniciar el desarrollo del algoritmo para la deformación de órganos.

Responsable: Programador.

Participante: Programador, Jefe de Equipo, Arquitectura y tecnologías.

Entradas:

- Documentación.

Salidas:

- Primera versión de la deformación de órganos.

Descripción:

En esta actividad se comienza a desarrollar un algoritmo que permite que los órganos se deformen cuando se produzca alguna colisión. Para realizar esta actividad se utiliza la investigación de las técnicas y algoritmos para la deformación que está en el repositorio. Al terminar esta implementación se aplican **Pruebas de unidad** antes de guardarlo en el repositorio, con el objetivo de detectar los posibles errores que puedan existir. Este tipo de prueba se describe con mayor profundidad y cómo aplicarla, en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"*.

Actividad: Acoplar el modelo físico con sistema de detección de colisiones.

Responsable: Programador.

Participante: Programador.

Entradas:

- Documento ERS.
- Documentación.
- Colisiones dinámicas.

Salidas:

- Modelo físico con sistema de detección de colisiones.

Descripción:

Con esta actividad se logra un módulo integrado a partir del módulo de detección de colisiones y el modelo físico masa-muelle que permite la generación del corte. A esta actividad se le aplican **Pruebas de caja negra**, para comprobar que el sistema satisface los requisitos del usuario, tanto los funcionales como los no funcionales; en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"* se describe con mayor profundidad y cómo realizar este tipo de prueba.

Actividad: Implementar el ejercicio para todos los niveles de complejidad definidos.

Responsable: Programador.

Participante: Programador, Líder de Proyecto, Jefe de Equipo, Arquitectura y tecnologías.

Entradas:

- Documento ERS.
- Diseño del ejercicio para todos los niveles de complejidad definidos.
- Generación de fluido.
- Simulación del hilo.
- Modelo físico con sistema de detección de colisiones.

Salidas:

- El ejercicio con tres niveles de complejidad.

Descripción:

A partir de las entradas que se tienen en esta actividad, se implementan los ejercicios correspondientes a esta etapa de desarrollo. Al terminar esta implementación se aplican **Pruebas de unidad** antes de guardarlo en el repositorio, con el objetivo de detectar los posibles errores que puedan existir. Este tipo de prueba se describe con mayor profundidad y cómo aplicarla, en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"*. Para la realización de estos ejercicios se utiliza la herramienta STK.

Actividad: Actualizar el módulo de gestión de la BD.

Responsable: Diseñador principal de base de datos.

Participante: Diseñador principal de base de datos, Programador.

Entradas:

- Diseño de la BD.
- El ejercicio con tres niveles de complejidad.

Salidas:

- Módulo de gestión de la BD.

Descripción:

Con la realización de esta actividad queda actualizado el módulo de gestión de BD ya que permite almacenar los diferentes valores que vayan tomando las variables de cada ejercicio. A esta actividad se le aplican **Pruebas de caja negra**, para comprobar que el mismo satisface los requisitos del usuario, tanto los funcionales como los no funcionales; en el trabajo de diploma titulado "*Propuesta de*

una estrategia de aseguramiento de la calidad para el SMQ" se describe con mayor profundidad y cómo realizar este tipo de prueba.

Actividad: Integrar los ejercicios de la etapa.
Responsable: Programador.
Participantes: Programador, Líder de Proyecto, Analista principal.
Entradas:

- Ejecutables de los ejercicios correspondientes a esta etapa.
- Descripción de la arquitectura del software.
- Documento ERS.
- La interfaz gráfica.

Salidas:

- Módulo conformado por los ejercicios de la etapa.

Descripción:
En esta actividad se crea el módulo donde se integran los ejercicios realizados a lo largo de la etapa de desarrollo. A esta actividad se le aplican **Pruebas de integración** de manera que se pueda comprobar que el módulo conformado por la integración de los ejercicios tuvo éxito, también se aplican **Pruebas que permitan validar las funcionalidades del sistema**, para comprobar que el mismo satisface los requisitos del usuario, tanto los funcionales como los no funcionales; en el trabajo de diploma titulado "*Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"* se describe con mayor profundidad y cómo realizar este tipo de prueba.

Actividad: Hacer una copia de salva.
Nota: Esta se realiza de la misma forma que en la etapa anterior.

Actividad: Elaborar el manual de usuario.
Nota: Esta se realiza de la misma forma que en la etapa anterior.

Actividad: Estudiar el concepto de simulador endoscópico.
Responsable: Investigador.
Participante: Investigador
Entradas:

Salidas:

- Documentación.

Descripción:

Investigar las características y funcionalidades que tienen los simuladores endoscópicos, así como su diseño gráfico para la construcción del mueble. La información que se obtiene es guardada en el repositorio.

Actividad: Revisión Técnica Formal.

Responsable: Planificador de la calidad.

Participante: Planificador de la calidad, Líder de Proyecto, Documentador, Investigador.

Entradas:

- Lista de chequeo.
- Glosario de términos.
- Documentación.

Salidas:

- Informe con los resultados obtenidos en la revisión.

Descripción:

Esta actividad permite realizar un chequeo a la actividad anterior para detectar los posibles errores que puedan existir teniendo en cuenta la lista de chequeo definida en el trabajo de diploma *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"*, esta lista permite medir la calidad del proceso.

Actividad: Estudiar elementos de sensado en el simulador endoscópico.

Responsable: Investigador.

Participante: Investigador.

Entradas:

- Documentación.

Salidas:

- Elementos de sensado.

Descripción:

Identificar, según las características de este tipo de simuladores y del proceso en sentido general, cuáles serían los elementos a sensar que aporten la información necesaria para simular el proceso.

Actividad: Revisión Técnica Formal.

Responsable: Planificador de la calidad.

Participante: Planificador de la calidad, Líder de Proyecto, Documentador, Investigador.

Entradas:

- Lista de chequeo.
- Glosario de términos.
- Elementos de sensado.

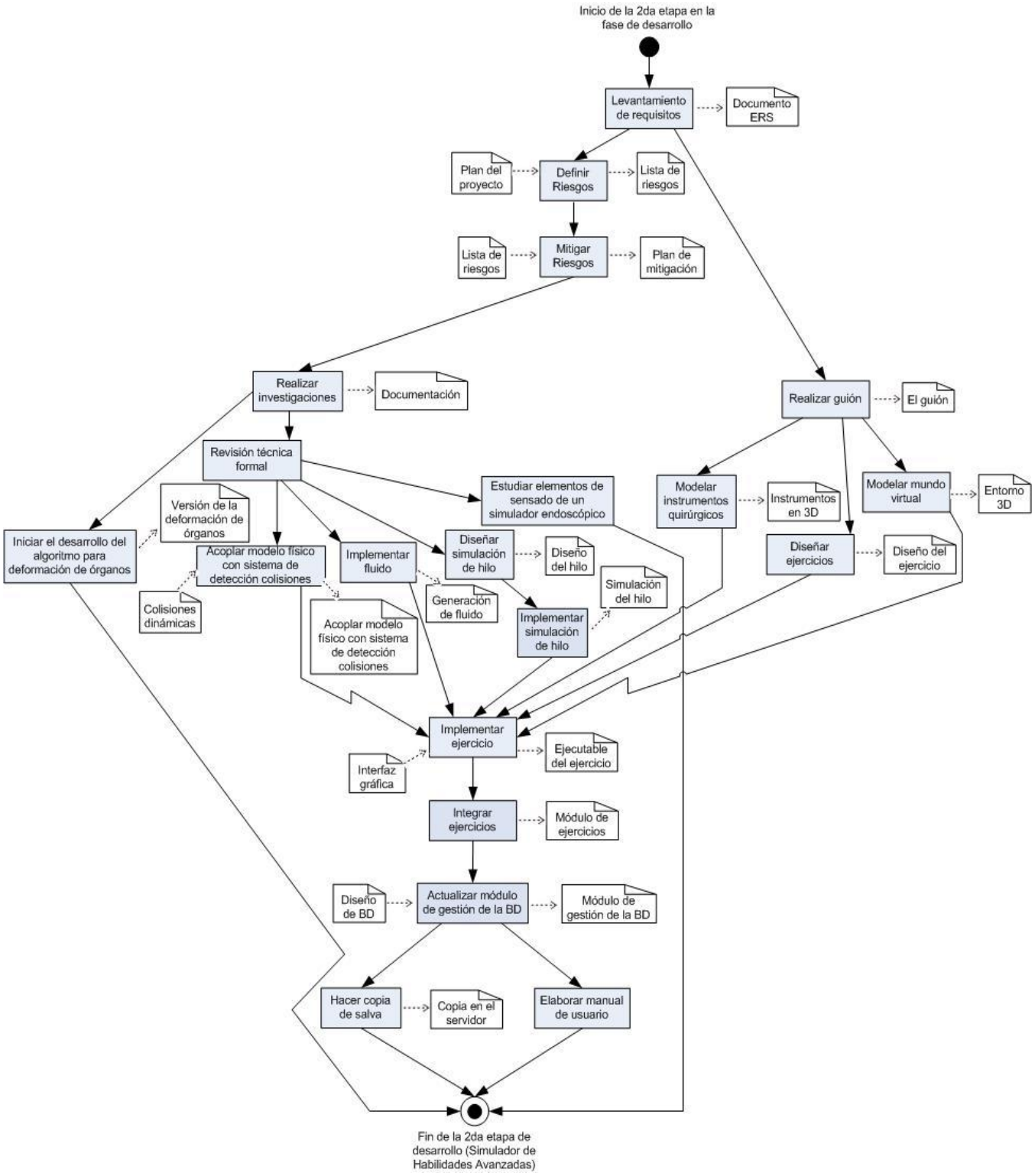
Salidas:

- Informe con los resultados obtenidos en la revisión.

Descripción:

Esta actividad permite realizar un chequeo a la actividad anterior para detectar los posibles errores que puedan existir teniendo en cuenta la lista de chequeo definida en el trabajo de diploma *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"*, esta lista permite medir la calidad del proceso.

En la siguiente figura se muestran las actividades que conforman la segunda etapa de desarrollo.



Actividades correspondientes a la TERCERA ETAPA de desarrollo del software:

Actividad: Introducir modelos para la reconstrucción de órganos.
Responsable: Programador.
Participante: Programador.
Entradas:

- Documentación.
- Guión técnico del ejercicio.

Salidas:

- Modelos para la reconstrucción de órganos en 3D.

Descripción:
Esta actividad permite crear un ambiente anatómico en la escena mediante la generación de modelos en 3D de órganos, una vez insertados los algoritmos correspondientes y necesarios.

Actividad: Implementar efectos de cauterización y dinámica de fluidos.
Responsable: Programador.
Participante: Programador, Arquitectura y tecnologías.
Entradas:

- Documentación.
- Generación de fluido.
- Guión técnico del ejercicio.

Salidas:

- Efectos de cauterización y dinámica de fluidos.

Descripción:
En esta actividad se implementan efectos de cauterización y dinámica de los fluidos. Al terminar esta implementación se aplican **Pruebas de unidad** antes de guardarlo en el repositorio, con el objetivo de detectar los posibles errores que puedan existir. Este tipo de prueba se describe con mayor profundidad y cómo aplicarla, en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ".*

Actividad: Iniciar el desarrollo de software para la reconstrucción y modelación de órganos de secciones anatómicas.

Responsable: Programador.

Participante: Programador, Jefe de equipo.

Entradas:

- Guión técnico del ejercicio.
- Modelos para la reconstrucción de órganos en 3D.
- Simulador de Habilidades Avanzadas.

Salidas:

- Una versión inicial de la reconstrucción y modelación de órganos

Descripción:

A partir de las entradas que constan en esta actividad se logra reconstruir y modelar órganos en un entorno 3D.

Actividad: Iniciar el proceso de simulación de la retroalimentación al tacto.

Responsable: Programador.

Participante: Programador, Jefe de equipo.

Entradas:

- Documentación.
- Guión técnico del ejercicio.

Salidas:

- Versión inicial de la simulación de la retroalimentación al tacto.

Descripción:

En esta actividad se comienza el proceso de simulación de la retroalimentación al tacto que debido a su complejidad se terminará en la siguiente etapa.

Actividad: Inicio del desarrollo del editor de propiedades físicas.

Responsable: Programador.

Participante: Programador, Arquitectura y tecnologías.

Entradas:

- Documentación.
- Guión técnico del ejercicio.

Salidas:

- Una primera versión del editor de propiedades físicas.

Descripción:

En esta actividad se comienza a desarrollar el editor de propiedades físicas que permite obtener modelos deformables con las propiedades definidas según el modelo físico a aplicar, para esta primera versión se utiliza la herramienta Visual Studio 2003 y la STK, de esta última un módulo para cuerpos rígidos y deformables. Hasta el momento este editor solo se aplica a cuerpos rígidos, como resultado de la misma se obtiene un módulo. En la siguiente etapa se continuará su desarrollo aplicándolo también a cuerpos deformables. Al terminar esta actividad se aplican **Pruebas de unidad** antes de guardarlo en el repositorio, con el objetivo de detectar los posibles errores que puedan existir. Este tipo de prueba se describe con mayor profundidad y cómo aplicarla, en el trabajo de diploma titulado "*Propuesta de una estrategia de aseguramiento de la calidad para el SMQ".*

Actividad: Obtener la primera versión del software.

Responsable: Programador.

Participantes: Programador, Líder de Proyecto, Analista principal.

Entradas:

- Documento ERS.
- La interfaz gráfica.
- Descripción de la arquitectura del software.
- Una primera versión del editor de propiedades físicas.
- Versión inicial de la simulación de la retroalimentación al tacto.
- Efectos de cauterización y dinámica de fluidos.
- Versión inicial de la reconstrucción y modelación de órganos.

Salidas:

- Primera versión del software.

Descripción:

En esta actividad se obtiene una primera versión del software, a la cual se le aplican **Pruebas de caja negra**, para comprobar que el mismo satisface los requisitos del usuario, tanto los funcionales como los no funcionales y **Pruebas de integración,** ambas antes de guardarlo en el repositorio; en el trabajo de diploma titulado "*Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"* se describe con mayor profundidad y cómo realizar este tipo de prueba.

Actividad: Actualizar el módulo de gestión de la BD.

Responsable: Diseñador principal de base de datos.

Participante: Diseñador principal de base de datos, Programador.

Entradas:

- Diseño de la BD.
- Primera versión del software.

Salidas:

- Módulo de gestión de la BD.

Descripción:

Con la realización de esta actividad queda actualizado el módulo de gestión de BD ya que permite almacenar los diferentes valores que vayan tomando las variables de cada ejercicio. A esta actividad se le aplican **Pruebas de caja negra**, para comprobar que el mismo satisface los requisitos del usuario, tanto los funcionales como los no funcionales; en el trabajo de diploma titulado "*Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"* se describe con mayor profundidad y cómo realizar este tipo de prueba.

Actividad: Hacer una copia de salva.

Responsable: Administrador y configuración.

Participante: Administrador y configuración, Programador.

Entradas:

- Primera versión del software.

Salidas:

- Copia de salva en el repositorio.

Descripción:

Esta actividad permite hacer una copia de salva de esta primera versión en el repositorio. Para realizar esta actividad es necesario tener instalada la herramienta *TortoiseSVN* que es utilizada para realizar el control de versiones.

Actividad: Elaborar el manual de usuario.

Responsable: Programador.

Participante: Programador, Líder de Proyecto, Jefe de equipo.

Entradas:

- Primera versión del software.

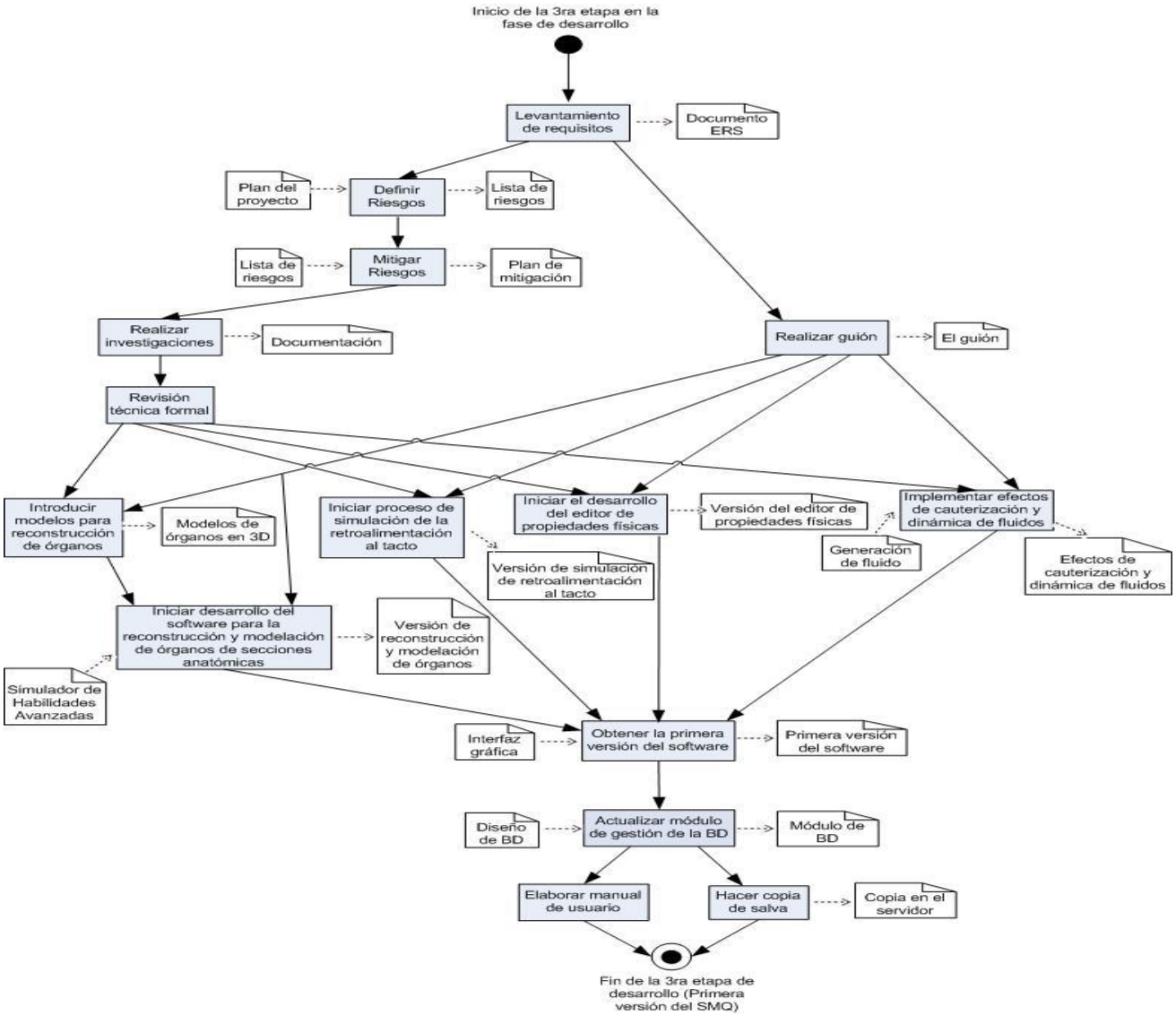
Salidas:

- El manual de usuario.

Descripción:

Se elabora un manual de usuario para esta etapa del proceso de desarrollo del SMQ donde se especifica detalladamente, paso a paso, cómo utilizar el sistema en cada una de las funcionalidades definidas en esta etapa de desarrollo. De esta forma el usuario podrá contar con una guía de apoyo en el momento de la ejecución del sistema.

En la siguiente figura se muestran las actividades que conforman la tercera etapa de desarrollo.



Actividades correspondientes a la CUARTA ETAPA de desarrollo del software:

Actividad: Terminar el software que permita la reconstrucción de órganos reales a partir de scanner.
Responsable: Programador.
Participante: Programador, Líder de Proyecto, Jefe de equipo.
Entradas:

- Guión técnico del ejercicio.
- Versión inicial de la reconstrucción y modelación de órganos.

Salidas:

- Software para la reconstrucción de órganos reales a partir de scanner.

Descripción:
En esta actividad se termina el desarrollo del software a partir de imágenes resultantes del scanner, el cual permite reconstruir y modelar órganos reales en un entorno 3D. Al terminar esta actividad se le aplican **Pruebas de unidad** antes de guardarlo en el repositorio, con el objetivo de detectar los posibles errores que puedan existir. Este tipo de prueba se describe con mayor profundidad y cómo aplicarla, en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ*".

Actividad: Finalizar el proceso de simulación de la retroalimentación al tacto.
Responsable: Programador.
Participante: Programador.
Entradas:

- Guión técnico del ejercicio.
- Versión inicial de la Simulación de la retroalimentación al tacto.

Salidas:

- Módulo de la retroalimentación al tacto.

Descripción:
Esta actividad permite culminar el proceso de simulación de retroalimentación de fuerza. Al terminar esta actividad se aplican **Pruebas de unidad** antes de guardarlo en el repositorio, con el objetivo de detectar los posibles errores que puedan existir. Este tipo de prueba se describe con mayor profundidad y cómo aplicarla, en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ".*

Actividad: Terminar el desarrollo del editor de propiedades físicas.

Responsable: Programador.

Participante: Programador, Arquitectura y tecnologías.

Entradas:

- Guión técnico del ejercicio.
- Una primera versión del editor de propiedades físicas.

Salidas:

- Editor de propiedades físicas.

Descripción:

En esta actividad se obtiene el editor de propiedades físicas aplicables a cuerpos rígidos y deformables. Al terminar esta actividad se aplican **Pruebas de unidad** antes de guardarlo en el repositorio, con el objetivo de detectar los posibles errores que puedan existir. Este tipo de prueba se describe con mayor profundidad y cómo aplicarla, en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ".*

Actividad: Obtener el ejecutable que tenga integrado el manejo de órganos adquiridos a partir de scanner y retroalimentación al tacto.

Responsable: Analista principal.

Participantes: Analista principal, Líder de Proyecto, Programador,

Entradas:

- Documento ERS.
- La interfaz gráfica.
- Descripción de la arquitectura del software.
- Software para la reconstrucción de órganos a partir de scanner.
- Módulo de la retroalimentación al tacto.
- Editor de propiedades físicas.

Salidas:

- Ejecutable que tenga integrado el manejo de órganos adquiridos a partir de scanner y retroalimentación al tacto.

Descripción:

En esta actividad se logra un ejecutable que tenga integrado el manejo de órganos adquiridos a partir de scanner y retroalimentación al tacto, a la cual se le aplican **Pruebas de caja negra**, para comprobar que satisface los requisitos del usuario, tanto los funcionales como los no funcionales y **Pruebas de integración,** ambas antes de guardarlo en el repositorio; en el trabajo de diploma titulado "*Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"* se describe con mayor profundidad y cómo realizar este tipo de prueba.

Actividad: Actualizar el módulo de gestión de la BD.
Responsable: Diseñador principal de base de datos.
Participante: Diseñador principal de base de datos, Programador, Arquitectura y tecnologías.
Entradas:

- Diseño de la BD.
- Ejecutable que tenga integrado el manejo de órganos adquiridos a partir de scanner y retroalimentación al tacto.

Salidas:

- Módulo de gestión de la BD.

Descripción:
Con la realización de esta actividad queda actualizado el módulo de gestión de BD ya que permite almacenar los diferentes valores que vayan tomando las variables de cada ejercicio. A esta actividad se le aplican **Pruebas de caja negra**, para comprobar que el mismo satisface los requisitos del usuario, tanto los funcionales como los no funcionales; en el trabajo de diploma titulado *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ"* se describe con mayor profundidad y cómo realizar este tipo de prueba.

Actividad: Hacer una copia de salva.
Responsable: Administrador y configuración.
Participante: Administrador y configuración, Programador.
Entradas:

- Software para la reconstrucción de órganos.
- Módulo de la retroalimentación al tacto.
- Editor de propiedades físicas.

Salidas:

- Copia se salva en el repositorio.

Descripción:

Esta actividad permite hacer una copia de salva de esta primera versión en el repositorio. Para realizar esta actividad es necesario tener instalada la herramienta *TortoiseSVN* que es utilizada para realizar el control de versiones.

Actividad: Elaborar el manual de usuario.

Responsable: Programador.

Participante: Programador, Líder de Proyecto, Jefe de equipo.

Entradas:

- Ejecutable que tenga integrado el manejo de órganos adquiridos a partir de scanner y retroalimentación al tacto.

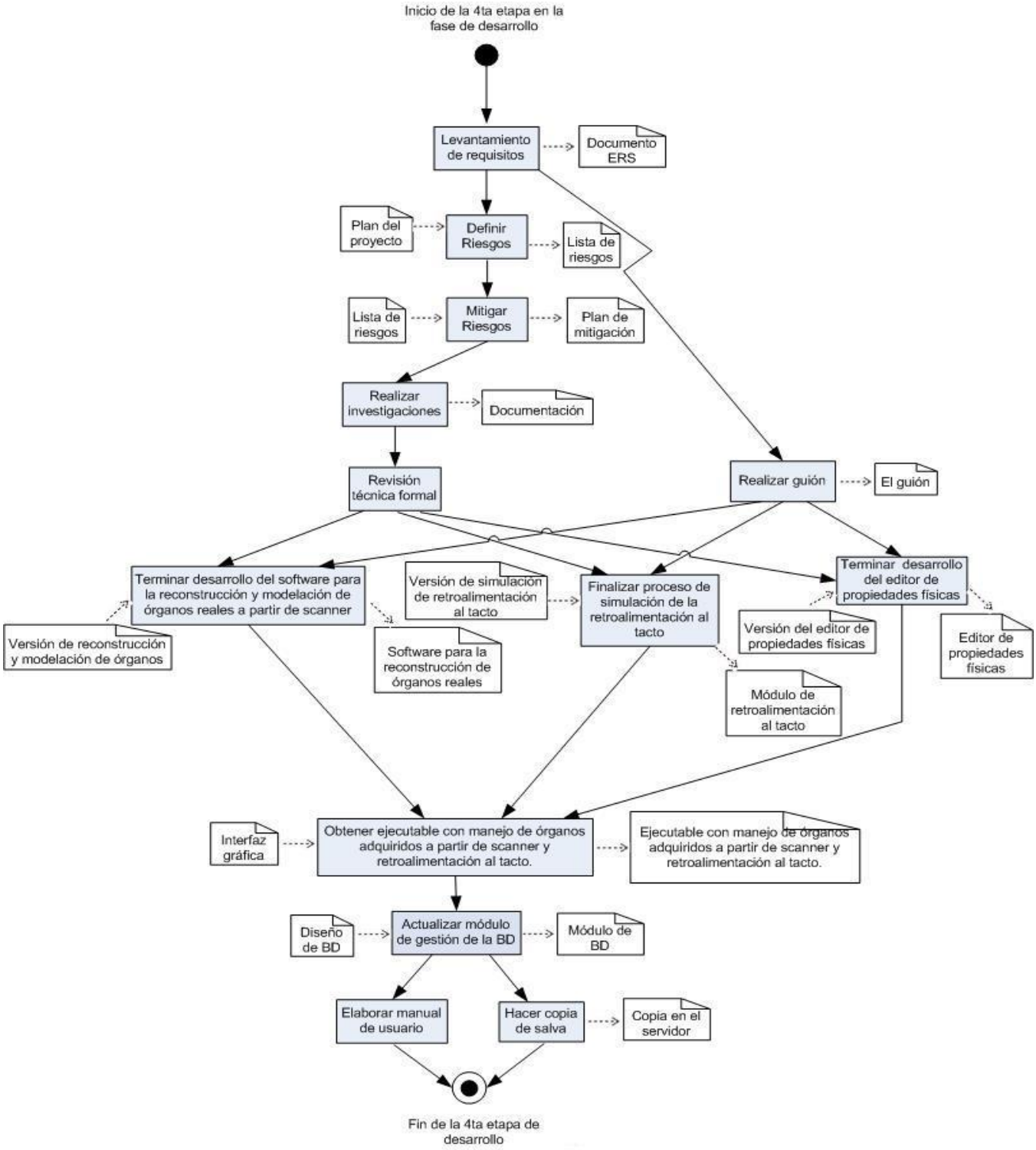
Salidas:

- El manual de usuario.

Descripción:

Se elabora un manual de usuario para esta etapa del proceso de desarrollo del SMQ donde se especifica detalladamente, paso a paso, cómo utilizar el sistema en cada una de las funcionalidades definidas en esta etapa de desarrollo. De esta forma el usuario podrá contar con una guía de apoyo en el momento de la ejecución del sistema.

En la siguiente figura se muestran las actividades que conforman la cuarta etapa de desarrollo.



FASE FINAL

Como se definió anteriormente, al inicio del proceso de desarrollo, el Producto KHEIPROS es un software a riesgo, pues no existe ningún cliente que solicitara la realización del mismo, sin embargo, a la hora de elaborar el proceso de desarrollo se tuvieron en cuenta las posibles actividades a desarrollar en caso que existiera cliente y de esta manera el proceso de desarrollo definido se podría emplear en la realización de otros Simuladores Quirúrgicos.

Esta última fase se realiza sobre la base de brindar solución a la fundamentación anterior; a continuación se describen las actividades a realizar en caso que exista al menos un cliente.

A petición del cliente se realizan o no **Pruebas de aceptación**, permitiendo de acuerdo a los requisitos planteados, comprobar la calidad del producto y el cumplimiento de sus expectativas. Con esta prueba el cliente valida el nivel de precisión que tiene el software. En caso que el cliente tenga incongruencias con el producto desarrollado, la próxima actividad a realizar sería **Solucionar problemas detectados por el cliente**.

Actividad: Instalar el producto en su entorno final.
Responsable: Líder de Proyecto.
Participante: Líder de Proyecto.
Entradas:

- El software.

Salidas:

- El Producto instalado.

Descripción:

Esta actividad permite que el producto resultante sea ubicado principalmente en instituciones médicas donde se practiquen cirugías laparascópicas.

Actividad: Entrega de documentación final.
Responsable: Líder de Proyecto.
Participante: Líder de Proyecto.
Entradas:

- El manual de usuario generado en cada etapa de desarrollo.

Salidas:

- Informe completo del proyecto.

Descripción:

Para realizar esta actividad se parte de los manuales de usuarios definidos en las etapas anteriores y a partir de ellos se confecciona un informe completo para lograr que los usuarios finales entiendan y puedan manipular de una forma sencilla el producto.

Actividad: Probar el producto en su entorno final.

Responsable: Analista principal.

Participante: Analista principal.

Entradas:

- Documento ERS.

Salidas:

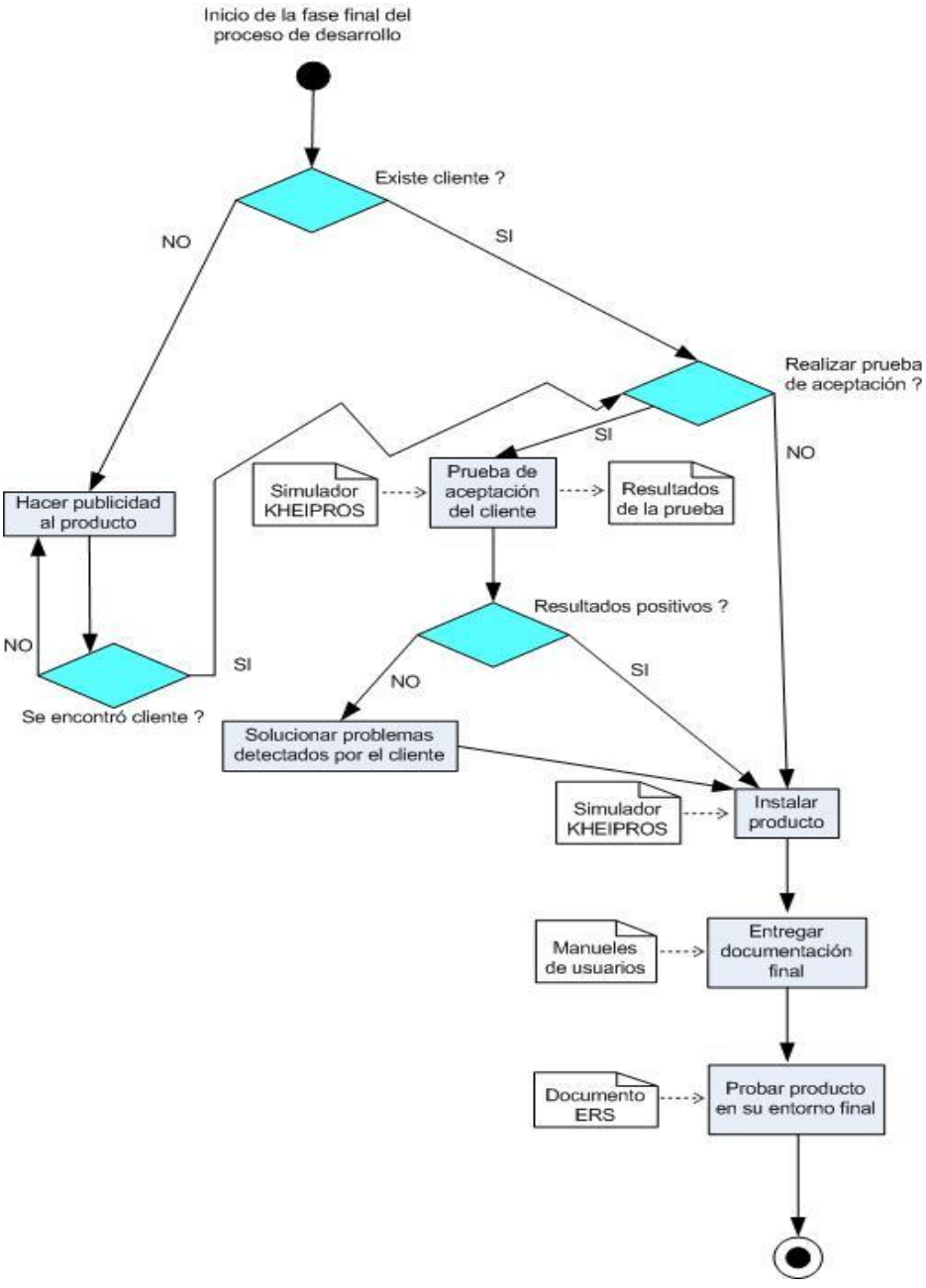
- Resultados de las pruebas.

Descripción:

Esta actividad garantiza que se realicen pruebas al producto una vez que esté fuera de su entorno de desarrollo, lo que permite medir el nivel de adaptabilidad del mismo en un nuevo entorno, es necesario además, que el cliente participe durante la realización de esta prueba.

En caso de no existir un cliente, se estaría en presencia de un proyecto a riesgo. En este caso las personas implicadas en el rol de Responsable de vigilancia tecnológica son los encargados de registrar a partir de un estudio del mercado sistemático, la información referente a cambios que puedan surgir mientras se esté desarrollando el producto en cuanto a: línea temática, tendencia de mercado y figuras líderes que se dedican a desarrollar este tipo de sistema. Esta información se le proporciona al grupo de colaboración-comercialización de la Universidad, que tiene como nombre “Campaña Comunicacional”, estos son los encargados de **Hacer publicidad al producto** mediante catálogos, videos, etc. Estos elementos de publicidad se hacen con el objetivo de difundir el producto en los nichos de mercado identificados.

A continuación se muestran las actividades que se desarrollan en la fase final:



Principales herramientas utilizadas en el SMQ

La herramienta TRAC se emplea para controlar en el proyecto SMQ las tareas que se le asignan a cada persona implicada en el mismo.

La herramienta *TortoiseSVN* se emplea para lograr una cohesión en el ejercicio que se esté desarrollando. Además permite fusionar los trabajos de cada uno de los programadores ya que estos pueden trabajar con diferentes clases del módulo y solo tienen que actualizar al terminar.

Se emplea la herramienta 3DMax para realizar el diseño gráfico y la herramienta Visual Paradigm para realizar el modelado visual.

Se emplea la herramienta Visual Studio 2003 para crear y compilar el código que garantiza las funcionalidades del sistema

CONCLUSIONES

Para desarrollar soluciones con calidad y profesionalismo en la industria del software, se hace imprescindible tener bien definido el proceso de desarrollo a seguir durante la producción y aún más cuando se trata de un producto que tiene alta repercusión en la sociedad, como es el caso del Simulador Quirúrgico que se está desarrollando en el proyecto SMQ de la Facultad 5, el cual será usado para el entrenamiento de las técnicas de Cirugía de Mínimo Acceso, de manera que el sistema garantice que los aprendices adquieran las habilidades necesarias para llegar a especializarse en el tema.

En este trabajo se definió el proceso de desarrollo a seguir en ese proyecto para desarrollar el Simulador Quirúrgico, dando cumplimento al objetivo planteado en esta investigación. Este proceso abarca desde la etapa de conceptualización del proyecto hasta la entrega del producto al cliente, está estructurado en tres fases para las cuales se define un flujo de actividades a seguir y estas a su vez tienen además definido su objetivo, las entradas y salidas y los roles participantes. También se identificaron los elementos de las metodologías de desarrollo del software estudiadas que contribuyen al perfeccionamiento del proceso y se integraron a este. Sin duda, el flujo de actividades propuesto para cada etapa del proceso va a mejorar la productividad de los equipos de desarrollo del proyecto.

Complementando la solución propuesta en este trabajo se diseñó la primera versión de la plantilla para elaborar el guión técnico del Simulador, con el objetivo de lograr incorporar cuanto antes este artefacto al proceso de desarrollo, dado que esto se detectó como una necesidad imperante en el proyecto.

Todo el empeño de esta investigación estuvo enfocado a disciplinar el desarrollo del Simulador Quirúrgico y se espera que al ser aplicada la solución propuesta se obtengan los resultados vislumbrados.

RECOMENDACIONES

Se considera necesario al culminar el presente trabajo hacer las siguientes recomendaciones:

- Poner en práctica el proceso de desarrollo definido, en el proyecto Simulador Quirúrgico.
- Elaborar el guión técnico del Simulador empleando la plantilla definida en esta investigación.
- Realizar investigaciones futuras para enriquecer el proceso definido a partir del análisis de otras metodologías de desarrollo del software que se puedan integrar a él.
- Realizar un estudio que permita proponer métricas específicas para medir la calidad del producto.
- Aplicar la métrica propuesta en el trabajo de diploma *"Propuesta de una estrategia de aseguramiento de la calidad para el SMQ*", para evaluar la eficacia del proceso definido.

REFERENCIA BIBLIOGRÁFICA

- [En línea] http://www.upv.es/crib/docs/ficha_ci2b_simrv.pdf.
- [En línea] http://diccionarios.elmundo.es/diccionarios/cgi/lee_diccionario.html?busca=cirugia+endoscopica&submit=+Buscar+&diccionario=8 .
- [En línea] http://es.wikipedia.org/wiki/Simulaci%C3%B3n.
- **2008.** [En línea] febrero 2008. http://www.latinsalud.com/articulos/00142.asp.
- **2002.** [En línea] 2002. http://www.immersion.com/.
- [En línea] http://www.egattaca.com/eContent/library/documents/DocNewsNo50DocumentNo6.PDF.
- [En línea] http://www.gpicr.com/msf.aspx.
- [En línea] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.
- **1998.** *Best Practices for Software Development Teams.* 1998.
- **2002.** *Rational Software .* 2002.
- **Beck, K.** *"Extreme Programming Explained. Embrace Change".*
- **Cotin, S, Delingette, H and Ayache, N. 1999.** *Real-time Elastic Deformations of Soft Tissues for Surgery Simulation.* 1999.
- **Fowler, M and Foemmel, M. 2001.** [Online] 2001. www.martinfowler.com/articles/designDead.html.
- **Jeffries, R, Anderson, A and Hendrickson, C. 2000.** *"Extreme Programming Installed".* 2000. 13: 9780201708424.
- **Kent Beck.** "Extreme Programming Explained". [Online] http://www.amazon.com/exec/obidos/ASIN/0201616416/programacione-20.
- **Kruchten, P. 2000 .** *The Rational Unified Process: An Introduction.* 2000 .
- **Parra Márquez, Juan Carlos, García Alvarado, Rodrigo and Santelices Malfanti, Iván. 2001.** *Introducción Práctica a la Realidad Virtual.* 2001.
- **Paulk, M., et al.,. 1993.** *Capability Maturity Model for Software.* 1993.
- **Pressman, Roger S.** *Ingenieria del Sotware. Un enfoque practico.* p. 19.
- **2001.** *Procesos del Ciclo de Vida del Software.TECNOLOGÍAS DE LA INFORMACIÓN.* 2001.
- **2008.** wikilearning.com. [Online] 2008. http://www.wikilearning.com/curso_gratis/metodologias_de_desarrollo_de_software-metodologia_de_desarrollo_de_software/3617-1.
- Wikipedia. [En línea] http://es.wikipedia.org/wiki/Simulaci%C3%B3n.

BIBLIOGRAFÍA

- [En línea] http://www.marblestation.com/blog/?p=644 .
- [En línea] http://www.scribd.com/doc/297224/RUP .
- [En línea] http://www.e-gattaca.com/eContent/NewsDetail.asp?ID=50&IDCompany=4 .
- [En línea] http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html .
- **1998 .** [En línea] 1998 . http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html#Jacobson.
- [En línea] http://www.gpicr.com/msf.aspx .
- [En línea] http://www.naturastock.com/rsdotnet/iic3140/materia/iic3140_04.pdf.
- [En línea] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/tema3_1xh.pdf.
- [En línea] http://www.informatizate.net .
- *kynetia.es.* [En línea] http://www.kynetia.es/calidad/rup-rational-unified-process.html.
- **Gomez. 2006.** [En línea] Mayo de 2006. http://drgomezsancha.blogspot.com/2006/05/simulador-de-realidad-virtual-para.html.
- **Meier, C y Monserrat, M.** SIMULACIÓN QUIRÚRGICA. [En línea] http://www.dsic.upv.es/~cmonserr/Articulos/Simul2003.pdf.
- seclaendosurgery. [En línea] www.seclaendosurgery.com.

ANEXOS

Anexo 1

Anexo 2

Anexo 3

Anexo 3.1

Anexo 4. Plantilla para elaborar el guión técnico del Simulador Quirúrgico

En este documento se especifican las ideas a desarrollar en el módulo, que será el trabajo de otros desarrolladores como programadores y diseñadores, por tanto, es necesario no dejar escapar ningún detalle. A continuación de muestran los elementos fundamentales que guían la confección de este documento.

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	*<x.x>*	*<detalles>*	*<nombre>*

Descripción del módulo

[Descripción del módulo del Simulador que se abordará en el documento, explicando de qué trata, cuáles son sus objetivos y funcionalidades, mencionar los aspectos interesantes y/o innovadores que tenga.]

Herramientas y lenguajes de desarrollo

[Se especifican los lenguajes en los que se programará y las herramientas que se emplearán durante el proceso de desarrollo del módulo. Los compiladores, las herramientas de diseño gráfico, las de gestión de la configuración, en fin, todo lo que se emplea para automatizar las tareas de desarrollo. Se debe especificar la ubicación donde estará almacena dicha herramienta para que todos los desarrolladores puedan acceder a ella.]

Herramienta	Descripción	Ubicación en las carpetas del repositorio
Microsoft Visual Studio 2005	*Será empleada para programar el módulo usando el lenguaje C++…*	*Tools\VS_2005*

Descripción de la maqueta de hardware

[Se describe la maqueta de hardware que será integrada al software. Es válido aclarar que esta descripción es igual para todos los módulos del Simulador, por eso en los otros guiones se puede referenciar. Aquí se debe detallar cómo es la interacción con el hardware y los controles que posee. El objetivo de este tópico es que los desarrolladores sean capaces de manipular el hardware correctamente con sólo leer esta explicación y así poder usarlo para hacer las pruebas a las funcionalidades que desarrollan. En caso de que el equipo tenga manual de usuario se debe especificar cómo acceder a él.]

Interfaz

[Mediante la interfaz se establece la comunicación entre el software y el usuario. Por tanto, es importante describir cada elemento de la pantalla, por ejemplo: el menú de opciones; los botones, el significado de las estadísticas que se muestran en el ejercicio, y cualquier otro elemento que permita interactuar con los ejercicios. En caso de que la interfaz varíe al desatarse determinado evento se deben especificar los cambios que ocurren y por supuesto, el evento que los provoca. Se recomienda anexar a este documento un prototipo visual de la interfaz de usuario para ayudar a su elaboración.]

Entornos

[Se trata de describir todos los escenarios con los objetos u órganos que lo componen, las luces que lo afectan, en fin, todos los detalles, de modo que permita a los diseñadores gráficos y programadores la confección del mismo. Se describe cada Entorno por separado y luego se resumen en la siguiente tabla. Una vez que el mismo haya sido diseñado se especifica su ubicación para que los programadores puedan acceder a él. Se recomienda anexar imágenes]

Entorno1……

Entorno 2……

Entorno	Descripción	Ubicación en las carpetas del repositorio
Tuvo digestivo	*Es el escenario del nivel 1 del ejercicio de Cámaras.*	*Proyecto/Entornos/Cámara1.3dx*

Instrumentos

[Se describen los instrumentos médicos que serán modelados para el módulo. La idea es dar todos los detalles posibles para guiar el trabajo de los diseñadores gráficos. Una vez modelados, se incluyen en la siguiente tabla. Se recomienda anexar imágenes]

Instrumento	Descripción	Ubicación en las carpetas del repositorio
Bisturí	*Instrumento que se usa para el corte.*	*Proyecto/Instrumentos/Bisturí.3dx*

Mecánicas de la simulación

[Aquí se deben describir perfectamente todos los efectos o respuestas que son causados por la simulación de determinado proceso, por ejemplo para el módulo donde se simula la cauterización se puede especificar que después de haber transcurrido 4 segundos de que el usuario esté realizando la cauterización el color del tejido en la zona cauterizada debe ponerse carmelita brillante y se debe visualizar la animación de humo sobre esa zona, otro ejemplo es en el ejercicio de selección que una vez que el usuario lograr interceptar un objeto de la escena con la pinza correspondiente, este objeto desaparece y en su posición se muestra una animación de explosión.

Es útil organizarlas por niveles, en el caso en que haya mecánicas que aparezcan sólo en determinados niveles.]

Ambiente sonoro

[En esta parte hay que describir cómo va a ser la ambientación sonora del módulo. Se debe especificar qué tipo de ambiente se quiere crear en cada proceso simulado, si es que lo lleva, y en cada caso especificar bien el momento de aparición y duración en la escena del sonido.

Cámaras

[Detallar como va a ser el manejo de la cámara, si el entorno tiene una cámara en tercera persona o una en primera persona, si es fija o se puede mover mediante controles. En cualquier caso hay que definir cómo va a ser.]

Inteligencia Artificial

[Se deben describir los comportamientos inteligentes del sistema. Por ejemplo, pudiera ser que durante la realización de un ejercicio se le muestre al usuario un mensaje alertando de un error que el sistema Determino que era muy probable que este cometiera]

Anexos

[Aquí se deben colocar fotografías y diseños de referencia que permitan complementar las explicaciones anteriores y ayuden a una mejor comprensión de las mismas. También sería bueno que se pusieran ilustraciones propias, croquis de un entorno, etc.]

Otras especificaciones

[Aquí se incluyen otras especificaciones que deban ser conocidas por los que programan el módulo y que no se hayan tratado en los tópicos anteriores.]

GLOSARIO DE TÉRMINOS

Cirugía artroscópica: operación en el interior de una articulación.

Cirugía cardiaca: operación al corazón.

Colecistectomía: extirpación quirúrgica de la vesícula biliar.

Disectores: instrumento quirúrgico con mango, articulado en el centro, que permite la disección de tejidos por el cirujano, al separarlos o atravesarlos. Suele tener punta curva, más o menos angulada, para separar los tejidos, como al abrir una tijera, o atravesarlos de forma roma cuando las patas del instrumento están cerradas.

Feedback: retroalimentación.

Framework: marco de trabajo.

Háptico: se refiere al contacto, al conjunto de sensaciones no visuales y no auditivas que experimenta un individuo.

Laparoscopía: técnica quirúrgica de visualización directa de las vísceras del abdomen sin abrirlo propiamente, que se realiza llenando la cavidad peritoneal (habitualmente virtual) de gas, para crear así un espacio en el que introducir percutáneamente una cámara conectada a un monitor de televisión.

Marketing: es el proceso social y administrativo por el cual los grupos e individuos satisfacen sus necesidades al intercambiar bienes y servicios, involucra estrategias de mercado, de ventas, estudio de mercado, posicionamiento de mercado, etc.

Mediastinoscopía: técnica de visualización del mediastino y obtención de biopsias de los ganglios linfáticos mediastínicos. Sus principales complicaciones son la hemorragia por biopsia y la parálisis de las cuerdas vocales por lesión del nervio recurrente.

Peritoneo: membrana serosa que recubre la pared de la cavidad abdominal (peritoneo parietal) y las vísceras intraperitoneales (peritoneo viceral). Entre ambas hojas peritoneales existe la cavidad peritoneal, que es virtual, pues en condiciones normales solo contiene unos mililitros de líquido peritoneal.

Reléase: una versión particular de un artículo de la configuración que se hace disponible para un propósito específico

Repositorio: lugar en el que se almacenan los datos actualizados e históricos, a menudo en un servidor.

Toracoscopía: observación de la cavidad pleural y obtención de muestras de esta mediante la introducción de un toracoscopio a través de la cavidad torácica. Se emplea, fundamentalmente, para facilitar el diagnóstico de derrames pleurales cuando la toracocentesis o la biopsia pleural a ciegas han sido insuficientes.

TortoiseSVN: es un cliente gratuito de código abierto para el sistema de control de versiones *Subversion*, maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en un *repositorio* central. El repositorio es prácticamente lo mismo que un servidor de ficheros ordinario, salvo que recuerda todos los cambios que se hayan hecho a sus ficheros y directorios. Esto permite que pueda recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo y cómo cambiaron sus datos, y quién hizo el cambio.

GLOSARIO DE ABREVIATURAS

ACPs: Áreas Clave de Procesos.

BD: Base de Datos.

CASE: Ingeniería de sistemas automatizados.

CMA: Cirugía de Mínimo Acceso.

CMM: Modelo de Capacidad de Madurez.

ERS: Especificación de Requisitos del Software.

IBM: International Business Machines.

IEC: Comisión Electrotécnica Internacional.

ISO: Organización Internacional para la Normalización.

MCV: Modelo de Ciclo de Vida.

MRI: Imagen de Resonancia Magnética.

RV: Realidad Virtual.

SEI: Instituto de Ingeniería de Software.

SIMPRO: Empresa dedicada al desarrollo de Simuladores Profesionales utilizando la Realidad Virtual.

SMQ: Simulador Quirúrgico.

SRV: Sistemas de Realidad Virtual.

SSQ: Sistema de Simulación Quirúrgico.

STK: Scene Toolkit.

TAC: Tomografía Axial Computarizada.

UCI: Universidad de las Ciencias Informáticas.

UML: Lenguaje de Modelado Unificado.

VS2005: Visual Studio 2005.