Universidad de las Ciencias Informáticas

Facultad 5 Entornos Virtuales

Estrategia de Pruebas para Juegos de Realidad Virtual

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autores:

Adriana Santos Lebeque

Danaisy Hernandez Valladares

Tutores:

Ing. Jandrich Dominguez Fortún

Ing. Gerandys Hernández Casanova

Ciudad de La Habana. Cuba

Junio de 2008

"Para ser exitoso no tienes que hacer cosas extraordinarias...

...Haz cosas ordinarias, extraordinariamente bien."

DECLARACIÓN DE AUTORÍA

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

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

_____ _____

Autor (a): Adriana Santos Lebeque Autor(a): Danaisy Hernandez Valladares

_____

Tutor: Ing. Jandrich Domínguez Fortún

_____

Tutor: Ing. Gerandys Hernández Casanova

Datos de Contacto

Autores:

Nombre: Adriana Santos Lebeque

Correo Electrónico: alebeque@estudiantes.uci.cu.

Nombre: Danaisy Hernandez Valladares

Correo Electrónico: dvalladares@estudiantes.uci.cu .

Tutores:

Nombre: Jandrich Domínguez Fortún.

Correo Electrónico: jandrich@uci.cu .

Ingeniero Industrial. Graduado en Julio de 2005. Desde este año trabaja en la Universidad de las Ciencias Informáticas como profesor de Matemática Aplicada.

Durante dos años se desempeñó como asesor de calidad de la Facultad 5 y en este propio período trabajó como responsable de calidad del Proyecto SCADA Nacional. Actualmente es asesor de la Dirección General de Producción. Cursó el diplomado de Docencia Universitaria y de Dirección, entre otros cursos de postgrado. Actualmente está cursando el último diplomado de la maestría de Gestión de Proyectos Informáticos. Ha participado con los resultados de sus investigaciones en numerosos eventos nacionales e internacionales entre los que se destacan: UCIENCIA 2006 y 2007, Forum de Ciencia y Técnica 2006 y 2007, 13 convención científica de ingeniería y arquitectura, Informática 2007, Octavo Congreso Nacional y Cuarto Internacional de red de la investigación y docencia sobre innovación tecnológica. Tiene varias publicaciones como memorias de eventos y en sitios Web temáticos. En el año 2007 le concedieron el sello forjadores del futuro y Premio de Rector al profesor universitario en adiestramiento más destacado. En este propio año alcanzó la categoría de docente Instructor. Cumplió Misión UCI en la República Bolivariana de Venezuela en los períodos 27/01/07-17/02/07 y 18/08/07-13/09/07 como parte del proyecto SCADA Nacional. Actualmente es Jefe del Grupo de Implantación de la Resolución 297/03 en la Infraestructura Productiva y se encuentra trabajando en el diseño del proceso productivo en la infraestructura Productiva.

Nombre: Gerandys Hernández Casanova.

Correo Electrónico: ghernandez@uci.cu.

Ingeniero en Ciencias Informáticas. Graduado en Julio de 2007. Desde este año trabaja en la Universidad de las Ciencias Informáticas como profesor de Administración de Empresa y Contabilidad y Finanzas. Se desempeña actualmente como líder del proyecto Calidad de la Facultad 5.

Asesor:

Nombre: José Manuel Pardo Matos

Correo Electrónico: jpardo@uci.cu

Ingeniero en Ciencias Informáticas. Graduado en Julio de 2007. Profesor del Departamento de Ciencias Básicas desde septiembre de 2007; Profesor Adiestrado. Asesor de Calidad de la Facultad 5. Actualmente cursando la maestría en Calidad de Software. Ha cursado varios postgrados durante su etapa de adiestramiento entre los que están Docencia e Innovación Universitaria, Auditoría TIC, Técnicas Avanzadas de desarrollo de SW, Métricas de SW, Validación y Verificación, Monitoreo y Control de Proyectos. Participó en UCIENCIA 2007.

Danaisy:

Dado el momento de agradecer, comenzaré diciendo que esta es una tarea sumamente difícil, puesto que son muchas las personas que han participado para ver el sueño de ser Ingeniera finalmente realizado, primero que todo me veo en la obligación de agradecerle a mis padres por todo el amor y el apoyo que me han dado, no solo a lo largo de estos cinco años sino en todo mi trayecto de vida, los quiero mucho, en un segundo plano pero para nada menos importante agradezco a mi hermano sus palabras , palabras que me han ayudado a levantar el animo en mas de una ocasión. Gracias a mis tías por su constante preocupación

De más esta decir que doy gracias por la oportunidad que tuve de estudiar en esta Universidad oportunidad que me permitió conocer compañeros que estuvieron presentes cuando añoraba la compañía de mi familia y el calor de mi hogar, puedo decir sinceramente que no soy capaz de describir lo mucho que los voy a extrañar. Gracias a todos los que me tendieron la mano en el que ha sido sin lugar a dudas el obstáculo más grande que tuve que atravesar en toda mi carrera (Las pruebas de Nivel), a mis tutores y en general a los que me ayudaron de una forma u otra con esta investigación.

Gracias!!!

Adriana:

Al comienzo me decía que este iba a ser el momento más difícil de mi tesis...y fue así, pues resulta extremadamente difícil agradecerle a cada una de las personas que de alguna forma han contribuido con este momento tan especial en mi vida,..pero voy a tratar de hacerlo lo mejor posible, sin mencionar nombres..para que no se me escape ninguno..

Quiero comenzar con Uds., mi mamá y mi papá...por nunca separarse de mi lado, por darme apoyo, por ser mis ejemplos de personas a seguir ,por ser inspiración y no rendirse jamás conmigo..a mi hermano, por sacar siempre lo mejor de mi y no perder nunca la Fe en su hermana mayor y a mi hermanita por impulsarme de forma inconsciente a ser mejor. Gracias a mis tíos y mis primos, los de aquí y los de allá... por estar siempre a mi lado, darme siempre los motivos para nunca rendirme y la ayuda para mirar siempre adelante, a mis queridos abuelos, por tener siempre listo ese consejo que me ayudara a mejorar como persona y a servirme de guía. Gracias a mi compañera de tesis, a mis tutores, todos mis amigos, y a todas esas personas que están cerca de mi corazón, Uds. han hecho posible que esta personita se convirtiera finalmente en una ingeniera...

Gracias!

A mami y papi, a mis hermanos...a mis abuelos, mis tíos, mis primos en fin a mi familia y a todas esas personas que han estado ahí, para mi..

Adriana

Dedicatoria

A mis padres, a mi hermano, a mi familia en general, por ser mi inspiración y mi apoyo en esta contienda

Danaisy

RESUMEN

Los juegos de Realidad Virtual constituyen un paso de avance tanto para la ciencia, como para la humanidad. Actualmente un tema de vital importancia en el mundo del software de juego, es la aplicación de pruebas a los mismos. Esta es una necesidad que surge a raíz del auge de entidades que producen y comercializan juegos a nivel mundial, lo que ha provocado un aumento significativo en la competitividad entre estos productores las cuales han elevado, en la misma medida, la calidad de los productos que comercializan.

La Universidad de Ciencias Informáticas (UCI) y en particular la Facultad 5, como entidad productora de software, desarrolla juegos de realidad virtual, y se ha visto en la necesidad inminente de realizar productos que se encuentren al nivel de la competitividad mundial, por lo que se hacía presente la necesidad de trazar una estrategia que comprendiera la planificación, diseño y ejecución de las pruebas en este tipo de juego.

PALABRAS CLAVE

Calidad, Calidad de Software, Estrategia de pruebas, Pruebas.

Índice

INTRODUCCIÓN

En las últimas décadas se ha experimentado un crecimiento vertiginoso a nivel mundial en la industria del software. En la actualidad, la mayoría de las empresas demandan automatizar procesos de vital importancia que son claves para su correcto funcionamiento, lo que ha desatado un gran auge de las empresas que producen y exportan software exitosamente a lo largo del mundo, unido a una competencia cada vez mas reñida en el mercado de productos de software.

Nuestro país, pese a los problemas económicos que enfrenta debido a las limitaciones que nos han sido impuestas, ha decidido insertarse en la Industria del software. Es así como el 23 de marzo del 2002 nuestro comandante indicó la creación de la Universidad de las Ciencias Informáticas (UCI) primera universidad surgida al calor de la Batalla de ideas. La UCI no solo es una escuela sino que también es una fábrica, pese a ser una institución joven, en la UCI se ha creado una infraestructura productiva en la que ya se han realizado productos de exportación y que se ha dado la tarea de formar personal altamente capacitado que produzca software de calidad con el objetivo de insertar a nuestro país en el mercado internacional.

Para lograr estos objetivos un factor indispensable es la calidad del software, el producto debe ser lo suficientemente bueno y cumplir con requerimientos de calidad para poder competir con productores de software que en muchas ocasiones son reconocidos.

En la UCI se imparten asignaturas obligatorias encaminadas a este tema, y se incluyen en los proyectos personal destinado a trazar una estrategia de pruebas que garantice la calidad del producto.

En el caso específico de los juegos de Realidad Virtual que se desarrollan en la Facultad 5, se han detectado algunos problemas a la hora de trazar dicha estrategia debido a la poca experiencia existente en proyectos de este tipo, la falta de personal calificado en aseguramiento y control de la calidad y las características que engloba un software de realidad virtual. Además de esto, no se cuenta con un estudio detallado de las pruebas que pueden ser adaptadas a este tipo de software, ni se cuenta con una organización de las mismas.

De aquí se infiere la necesidad de poner en práctica las pruebas de software dentro del proyecto desde el comienzo del desarrollo del mismo, pues mientras más rápido se apliquen

durante el ciclo de vida del software, más rápido se detectarán los problemas asociados con las mismas, se evitarían los costos de corregir errores y se lograría que el producto estuviese listo en el tiempo previsto bajo el presupuesto planeado.

La calidad de un software puede ser determinada viendo hasta que punto el producto realiza de forma adecuada las funciones para las que fue concebido. Para el software de juego, esto incluye la calidad de experiencia del jugador, qué tan adecuadamente las características del juego están implementadas y la calidad de las pruebas realizadas. Diversas actividades pueden ser realizadas para evaluar, medir, y mejorar la calidad de juego, las cuales son esenciales para lograr un producto final con calidad. Con estos fines se realiza la investigación cuyo **Problema Científico** es:

¿Cómo mejorar el proceso de pruebas en los juegos de Realidad Virtual de la Facultad 5?

El Objeto de Estudio de la investigación es el Proceso de Desarrollo del Software de Juegos y su Campo de Acción la Etapa de pruebas. En aras de dar solución al problema anterior se trazó como **objetivo general**:

Definir una estrategia que posibilite el correcto desarrollo de las pruebas a los juegos de Realidad Virtual, en la Facultad 5.

Para dar cumplimiento al objetivo se plantean los siguientes **Objetivos Específicos**:

- Investigar la teoría existente acerca de las pruebas a los juegos.
- Seleccionar y caracterizar los proyectos que desarrollan software de juegos en la en la Facultad 5.
- Seleccionar los tipos de pruebas que se le aplicarán al software.
- Determinar en qué momento de la vida del software serán aplicadas las pruebas seleccionadas.

Idea a defender:

Con la propuesta de una estrategia de pruebas se puede mejorar el proceso de pruebas en los juegos de Realidad Virtual.

Métodos de Investigación:

Para el desarrollo de la investigación científica, se utilizaron Métodos Teóricos y Empíricos que permiten analizar la información referente al tema, sintetizarla y llevarla al caso particular que se estudia con el objetivo final de arribar a una solución. Para el desarrollo de la fundamentación teórica los Métodos utilizados son:

- Analítico -sintético(Teórico)
- Inductivo-Deductivo(Teórico)
- Entrevista(Particulares)

Aportes prácticos esperados en el trabajo:

Con el desarrollo de este trabajo se espera obtener una estrategia de prueba que organice la estructura interna en cuanto a la aplicación de pruebas a los juegos de Realidad Virtual, con el objetivo de lograr que el software cumpla con las normas de calidad requeridas.

Breve referencia de cada capítulo

La tesis está estructurada en 3 capítulos:

Capítulo 1: Se describe de la situación problémica y la posible vía para darle solución además de realizarse un estudio acerca de las principales metodologías, herramientas y conceptos del proceso de pruebas.

Capítulo 2: En este capítulo se expone la situación por la que atraviesan los distintos proyectos que forman parte de nuestro objeto de estudio.

Capítulo 3: Se hace una propuesta de solución para el problema científico.

CAPITULO 1: FUNDAMENTOS TEÓRICOS

INTRODUCCIÓN

El estudio de las fuentes bibliográficas permite profundizar en los aspectos investigativos, así como conocer algunos criterios y valoraciones que abordan diferentes autores sobre una temática dada. Este análisis resulta de gran importancia, pues constituye los pilares necesarios para la realización de toda investigación científica.

El marco teórico de la investigación que a continuación se desarrolla en este capítulo está dirigido fundamentalmente, a destacar los aspectos más significativos abordando los principales conceptos, términos, metodologías y estrategias relacionadas con las pruebas de software, con el objetivo de lograr una mejor calidad en el software de juego que se desarrolla en la universidad.

1.1 CALIDAD

Aunque la palabra calidad tiene connotaciones distintas según las personas que la empleen, en ellas subyace siempre una idea central. La calidad de un producto es satisfactoria cuando responde a las necesidades del consumidor. La mayoría de los autores consideran que la calidad debe ser definida desde el punto de vista del cliente, reafirmando este planteamiento Feigenbam opina que la calidad es “Satisfacción de las expectativas del cliente”, igualmente Jurán plantea que “Calidad es adecuación al uso del cliente”. Por otra parte, La Real Academia de la Lengua Española plantea que la calidad es una “Propiedad o conjunto de propiedades inherentes a una cosa que permiten apreciarla como igual, mejor o peor que las restantes de su especie”. Donde las propiedades son características tales como: color, tamaño, forma, etc.

La calidad puede aplicarse, medirse o comprobarse en cualquier producto o servicio, la medida de la calidad puede ser vista a través de las diferencias obtenidas en la producción de estos, aunque no parezca cierto, siempre existen diferencias entre dos productos cualquieras, aunque sean dos de los mismos, muchos autores se refieren a esto como fenómeno de la varianza y afirman que sucede hasta en los procesos de la naturaleza. Para que un producto tenga una buena calidad es necesario disminuir al máximo estas diferencias.

A continuación se ofrecen otras definiciones de asociaciones y expertos reconocidos en el mundo de la calidad.

- Crosby: ”Calidad es cumplimiento de requisitos”
- Taguchi: “Calidad es la menor pérdida posible para la sociedad”.
- Deming: “Calidad es satisfacción del cliente”.
- Shewart:”La calidad como resultado de la interacción de dos dimensiones: dimensión subjetiva (lo que el cliente quiere) y dimensión objetiva (lo que se ofrece).
- La ISO 9000 define Calidad como “Grado en el que un conjunto de características inherentes cumple con los requisitos”.
- “El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas”. (1)

Teniendo en cuenta las definiciones expuestas anteriormente se puede concluir afirmando que se obtiene un producto con calidad cuando el mismo:

1. Cumple el propósito para el cual fue concebido.
2. Satisface las expectativas de los clientes y usuarios finales.
3. Es producido de acuerdo al tiempo previsto, y al presupuesto estimado para el mismo.

Si se pretenden aplicar estas definiciones al software el proceso se dificultaría, no se puede caracterizar al software con la misma facilidad con la que se caracteriza a un objeto físico.

El aumento de los productores de software y la complejidad del mismo, dio lugar a un mercado más competitivo en el que la calidad comenzó a ser un punto primordial en la construcción del software.

Años atrás la calidad era garantizada por los mismos desarrolladores, hoy en día esta se garantiza a través de equipos independientes que se rigen por metodologías, normas y estándares.

1.2 CALIDAD DE SOFTWARE

Anteriormente se estuvo hablando de calidad de software, debido a su importancia para el correcto desarrollo y funcionamiento de un software, este tema se ha convertido en uno de los principales problemas con los que se han enfrentado los productores de los mismos y el cual ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de software, es por ello que muchos han dirigido sus mayores esfuerzos a investigar entorno a:

1. ¿Cómo obtener un software con calidad?
2. ¿Cómo evaluar la calidad del software?

Como resultado de estas investigaciones muchos han sido los conceptos que se han derivado del término Calidad del software, entonces cabe preguntarse, ¿Qué es calidad de software?:

"La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario" (2)

El reconocido autor Roger S. Pressman, una autoridad internacionalmente reconocida en la mejora de procesos de software y en tecnologías de Ingeniería de software, define calidad de software como:

"Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente". (3)

Mas adelante modifica esta definición y plantea que la calidad es:

"Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario (4)

En opinión de los autores esta última definición es más abarcadora, ya que, en muchos casos, las necesidades que se plasman en el proyecto no reflejan lo que el consumidor desea en realidad. No se trata de que el consumidor no sepa lo que quiere, sino que no es capaz de expresar sus deseos en términos que resulten inteligibles para el equipo de desarrollo del proyecto.

1.3 ASEGURAMIENTO DE LA CALIDAD DE SOFTWARE

El Proceso de Aseguramiento de la Calidad es un proceso para proporcionar una adecuada seguridad de que los productos de software y los procesos en el ciclo de vida del proyecto están conformes con sus requisitos específicos y se ajustan a sus planes establecidos. Para que el aseguramiento de la calidad sea imparcial se necesita que tenga una libertad organizacional y una autoridad con respecto a las personas directamente responsables por el desarrollo del producto de software o la ejecución del proceso en el proyecto. El aseguramiento de la calidad puede utilizar los resultados de otros procesos de apoyo, tales

como el de Verificación, Validación, Revisiones Conjuntas, Auditorías y Solución del Problema. (5)

El aseguramiento de la calidad abarca todas aquellas actividades o prácticas que se realizan con el objetivo de asegurar un cierto nivel de calidad en el producto desarrollado que se realizan de forma independiente al equipo de desarrollo. (3)

“La garantía de calidad del software es un conjunto de procedimientos, técnicas y herramientas, aplicados por profesionales, durante el ciclo de desarrollo de un producto, para asegurar que le producto satisface o excede los estándares o niveles de calidad preestablecidos” (2)

Como se puede observar en algunos casos se le llama garantía de calidad y no aseguramiento, esto puede traer confusión con el término garantía del producto, por lo que es más recomendable referirse a ello como aseguramiento de la calidad.

El aseguramiento de la calidad se tiene en cuenta en métodos y herramientas de análisis, diseño, programación y prueba, así como en inspecciones técnicas formales en todos los pasos del proceso de desarrollo del software, además en las estrategias de prueba multiescala, control de la documentación del software y los cambios realizados, procedimientos para ajustarse a los estándares (y dejar claro cuando se está fuera de ellos), mecanismos de medida y también en los registro de auditorías y realización de informes . (6)

Existen tres aspectos muy importantes con relación al aseguramiento de la calidad del software (7):

- La calidad no se puede probar, se construye.
- El aseguramiento de la calidad del software no es una tarea que se realiza en una fase particular del ciclo de vida de desarrollo.
- Las actividades asociadas con el aseguramiento de la calidad del software deben ser realizadas por personas que no estén directamente involucradas en el esfuerzo de desarrollo.

Pressman considera que el aseguramiento de la calidad del software comprende una gran variedad de tareas asociadas (4):

- Preparar un plan de aseguramiento de la calidad del software para un proyecto.

- Participar en el desarrollo del proceso de descripción del proyecto de software.
- Revisar las actividades de ingeniería del software para verificar su consistencia con el proceso de software definido.
- Auditar el producto de software para verificar el cumplimiento del proceso de software definido.
- Asegurar que las divergencias en el trabajo de software sean documentadas de acuerdo a los estándares definidos. (6)
- Almacenar cualquier inconformidad y reportarla a la gerencia media.

El aseguramiento es imprescindible para tener confianza en que el software cumplirá con los requisitos de calidad. Son muchos los criterios que existen sobre como asegurar la calidad del software, pero todos concuerdan en que el aseguramiento se debe realizar desde el comienzo del ciclo de vida del producto. Se plantea que la prevención de los fallos implica menos costos que su corrección.

1.4 CONTROL DE LA CALIDAD

Entre las actividades que se realizan durante la elaboración de un producto se encuentra la del control de la calidad, cuyo propósito es el de asegurar la continua satisfacción de los clientes, ya sean externos o internos, mediante el constante monitoreo de la calidad del producto y los servicios que brinda el mismo. El control de la calidad facilita a la dirección un conjunto de técnicas y procedimientos para orientar, supervisar y controlar el proceso de desarrollo del software.

Muchos son los autores que han abundado sobre este tema, y se han enunciado conceptos tales como:

“El control de calidad es una serie de inspecciones, revisiones y pruebas utilizados a lo largo del proceso del software para asegurar que cada producto cumple con los requisitos que le han sido asignados.” (8)

De aquí se infiere que el control de la calidad se centra en cumplir las especificaciones elaboradas por el ingeniero del producto, a partir de los requerimientos de calidad propuestos por el cliente.

Según Monsalve (9) , el control de la calidad se relaciona con la vigilancia permanente de todo el proceso de desarrollo y el ciclo de vida del software. Se logra mediante la observación

constante del cumplimiento de cada una de las fases y actividades involucradas en el proceso de desarrollo.

- El control de la calidad permite realizar las rectificaciones necesarias a cualquier falla encontrada durante el proceso de desarrollo.
- Asegurar la calidad en las primeras fases del proceso de desarrollo del software implica que los costos del control en las etapas posteriores tiende a disminuir al tener menos aspectos que controlar, además de que la calidad estaría asegurada en sus bases.
- Para realizar un control de calidad deben ejecutarse frecuentes inspecciones a las metodologías de trabajo y el uso de las herramientas, revisiones de prototipos y de las pruebas formales de los productos finales. (9)

Como se puede observar una parte esencial del control de la calidad la juegan las pruebas del software.

1.5 PRUEBAS DE SOFTWARE

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa. Involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados, es por eso que la realización de las mismas a los software es un factor de vital importancia. Antes de liberar el producto es necesario tener la certeza de que este se encuentra libre de errores, aunque se considera que no se puede estar 100% seguro de esto. Pero una cosa si es cierta, mientras mas conocimiento se tenga del tema y más pruebas se le realicen al software más cerca se podrá estar de lograr un producto con la calidad esperada por los usuarios. De las pruebas se plantean varias normas que pueden servir acertadamente como objetivos de las mismas.

- Probar es un proceso de ejecución de un programa con la intención de descubrir un error.
- Una prueba tiene éxito si se descubre un error no detectado hasta entonces. (10)

De manera general el **objetivo** de la etapa de pruebas es garantizar la calidad del producto desarrollado, y para lograr esto se hace necesario:

1- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Las pruebas de integración son necesarias para

cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.

2- Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, creando componentes de pruebas ejecutables para automatizar las pruebas.
3- Realizar las diferentes pruebas y manejar los resultados de cada una de forma sistemática. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados. (11)

1.5.1 PROBLEMAS ASOCIADOS CON LAS PRUEBAS.

Antes de comenzar a diseñar y realizar las pruebas, es necesario que el personal destinado a estas tareas tenga la capacitación requerida, debido a que la falta de conocimiento a la hora de ejecutar las pruebas es un factor que conlleva al fracaso de las mismas.

Un problema que se encuentra estrechamente asociado con las pruebas es el fallo a la hora de definir correctamente los objetivos de estas, igualmente, realizar una prueba en una fase incorrecta del ciclo de vida del software entorpecería en gran medida la ejecución de las mismas, así como imposibilitaría la detección de muchos errores. Otro aspecto que se debe tener en cuenta es la técnica a utilizar para cada prueba, pues la aplicación de una técnica no efectiva a una prueba no cumple con sus objetivos, y además consume recursos y aumenta el costo de las mismas. (12)

1.5.2 TIPOS DE PRUEBAS

1.5.2.1 PRUEBAS UNITARIAS

Es la escala más pequeña de la prueba, está basada en la funcionalidad de los módulos del programa, como funciones, procedimientos, módulos de clase, etc. En ciertos sistemas también se verifican o se prueban los drivers y el diseño de la arquitectura. (8)

Los casos de pruebas que se diseñen para este nivel de pruebas, deben descubrir errores como:

- Comparaciones entre tipos de datos distintos.
- Operadores lógicos o procedencia incorrecta.
- Igualdad esperada cuando los errores de precisión la hacen poco probable.

- Las variables o comparaciones incorrectas.
- Terminaciones de bucles inapropiadas o inexistentes.
- Fallo de salida cuando se encuentra una iteración divergente.
- Bucles que manejan variables modificadas de forma inapropiada.

Para probar los componentes implementados como unidades individuales, se realizan las pruebas de especificación o de caja negra, que verifican el comportamiento de la unidad observable externamente y pruebas de estructura, o de caja blanca, que verifican la implementación interna de la unidad

1.5.2.2 PRUEBAS DE INTEGRACIÓN

Una vez que se les ha hecho la prueba de unidad a todos los módulos, nos preguntamos lo siguiente: “Si todos funcionan bien por separado, ¿por qué dudar de que funcionen todos juntos?”. Por supuesto, el problema es “ponerlos juntos” (interacción). Los datos se pueden perder en una interfaz; un módulo puede tener un efecto adverso e inadvertido sobre otro; las subfunciones, cuando se combinan, pueden no producir la función principal deseada; la imprecisión aceptada individualmente puede crecer hasta niveles inaceptables; y las estructuras de datos globales pueden presentar problemas; desgraciadamente, la lista sigue y sigue. La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción.

La mayoría de los casos de prueba de integración pueden ser derivados de las realizaciones de caso de uso-diseño, ya que las realizaciones de casos de uso describen como interaccionan las clases y los objetos, y por tanto como interaccionan los componentes. (11)

Se conocen dos tipos de integración, incremental y no incremental y aunque la selección de una estrategia de integración depende de las características del software y de la planificación del proyecto, en la mayoría de los casos existe la tendencia a aplicar la integración no incremental, es decir, a combinar todos los módulos por anticipado. Esto consiste en probar todo el programa en conjunto, lo que conduce normalmente al caos, debido a que se encuentra una gran cantidad de errores y la corrección de los mismos se hace difícil, puesto que es complicado aislar los errores al tener delante al programa completo. Una vez que se detectan esos errores aparecen otros nuevos y el proceso se repite en lo que parece ser un ciclo infinito.

Se aplica integración incremental cuando el programa se construye y se prueba en pequeños segmentos. Este tipo de integración tiene como ventajas que los errores son más fáciles de aislar y corregir, es más probable que se pueda probar completamente las interfaces y se puede aplicar un enfoque de prueba sistemática.

Existen dos estrategias de integración incremental:

- Integración Descendente
- Integración Ascendente

1.5.2.2.1 Integración descendente

La prueba de integración descendente es un planteamiento incremental a la construcción de la estructura de programas. Se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el módulo de control principal (programa principal).

1- Se usa el módulo de control principal como controlador de la prueba, disponiendo de resguardos para todos los módulos directamente subordinados al módulo de control principal.
2- Dependiendo del enfoque de integración elegido (primero-en-profundidad o primero-en-anchura) se van sustituyendo los resguardos subordinados uno a uno por los módulos reales.
3- Se llevan a cabo pruebas cada vez que se integra un nuevo módulo.
4- Tras terminar cada conjunto de pruebas, se reemplaza otro resguardo con el módulo real.
5- Se hace la prueba de regresión para asegurarse de que no se han introducido errores nuevos.

El programa continúa desde el paso 2 hasta que se haya construido la estructura del programa entero.

Para llevar a cabo las pruebas de integración descendentes es necesario crear resguardos, los cuales son una serie de programas que reemplazan los módulos de bajo nivel. Esto se hace necesario cuando se requiere un proceso de los niveles más bajos de la jerarquía para poder probar adecuadamente los niveles superiores.

Para darle solución a este problema se tienen tres opciones (8):

1- Retrasar muchas de las pruebas hasta que los resguardos sean reemplazados por los módulos reales.
2- Desarrollar resguardos que realicen funciones limitadas que simulen los módulos reales.
3- Integrar el software desde el fondo de la jerarquía hacia arriba.

1.5.2.2.2 Integración ascendente

La prueba de la integración ascendente, como su nombre indica, empieza la construcción y la prueba con los módulos atómicos (es decir, módulos de los niveles más bajos de la estructura del programa).

Se puede implementar una estrategia de integración ascendente mediante los siguientes pasos (3):

1- Se combinan los módulos de bajo nivel en grupos que realicen una subfunción específica del software.
2- Se escribe un controlador (un programa de control de la prueba) para coordinar la entrada y la salida de los casos de prueba.
3- Se prueba el grupo.
4- Se eliminan los controladores y se combinan los grupos moviéndose hacia arriba por la estructura del programa.

A medida que la integración progresa hacia arriba se hace menos necesario el uso de los controladores de prueba.

1.5.2.2.3 Prueba de regresión

La prueba de regresión consiste en volver a ejecutar un subconjunto de pruebas que se han llevado a cabo anteriormente para asegurarse de que los cambios no han propagado efectos colaterales no deseados.

1.5.2.2.4 Prueba de humo

La prueba de humo es un método de prueba de integración comúnmente utilizada cuando se ha desarrollado un producto software empaquetado. Es diseñado como un mecanismo para proyectos críticos por tiempo, permitiendo que el equipo de software valore su proyecto sobre una base sólida.

1.5.2.3 PRUEBAS DEL SISTEMA

La prueba del sistema, realmente, está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas.

Entre las pruebas de sistemas se pueden considerar las siguientes:

1.5.2.3.1 Prueba de recuperación

La prueba de recuperación es una prueba del sistema que fuerza el fallo del software de muchas formas y verifica que la recuperación se lleva a cabo apropiadamente.

1.5.2.3.2 La prueba de seguridad

La prueba de seguridad intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán, de hecho, de accesos impropios.

1.5.2.3.3 La prueba de resistencia

La prueba de resistencia ejecuta un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales.

1.5.2.3.4 La prueba de rendimiento

La prueba de rendimiento esta diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado.

Las pruebas de sistema se usan para probar que el sistema funciona correctamente como un todo. Cada prueba de sistema prueba principalmente combinaciones de casos de uso instanciados bajo condiciones diferentes. Estas condiciones incluyen diferentes configuraciones hardware (procesadores, memoria principal, discos duros, etc.) diferentes

niveles de carga del sistema, diferentes números de actores y diferentes tamaños de la base de datos. (11)

1.5.2.4 PRUEBAS DE ACEPTACIÓN

Estas pruebas se realizan para permitir que el cliente valide todos los requisitos. Las realiza el usuario final en lugar del responsable del desarrollo del sistema, una prueba de aceptación puede ir desde un informal hasta la ejecución sistemática de una serie de pruebas bien planificadas. De hecho, la prueba de aceptación puede tener lugar a lo largo de semanas o meses, descubriendo así errores acumulados que pueden ir degradando el sistema.

1.5.2.4.1 Pruebas Alfa y Beta

Es un proceso que llevan a cabo los desarrolladores para descubrir errores que parezca que solo el usuario final puede descubrir. La prueba *alfa* se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y los problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado. La prueba *beta* se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente. Así, la prueba beta es una aplicación en vivo del software en un entorno que no puede ser controlado por el desarrollador (8).

1.5.3 MÉTODOS DE PRUEBAS

1.5.3.1 Pruebas de Caja negra

Las pruebas de caja negra, también denominadas pruebas de comportamiento o de funcionalidad se centran es los requisitos funcionales del software, o sea, la prueba de caja negra permite al ingeniero de software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra intenta encontrar errores de las siguientes categorías (8):

1- Funciones incorrectas o ausentes.
2- Errores de interfaz.
3- Errores en estructuras de datos o en accesos a Bases de Datos externos.
4- Errores de rendimientos.
5- Errores de inicialización y de terminación.

1.5.3.2 Pruebas de Caja blanca

Las pruebas de caja blanca se centran en la estructura de control del programa. Se obtienen casos de prueba que aseguren que durante la prueba se han ejecutado, por lo menos una vez, todas las sentencias del programa y que se ejercitan todas las condiciones lógicas. Son conocidas también como estructurales o de cobertura lógica, en ellas se pretende indagar sobre la estructura interna del código, omitiendo detalles referidos a datos de entrada o salida. Para esta prueba se consideran tres puntos importantes (8):

1. Conocer el desarrollo interno del programa, determinante en el análisis de coherencia y consistencia del código.
2. Considerar las reglas predefinidas por cada algoritmo.
3. Comparar el desarrollo del programa en su código con la documentación pertinente. La primera parte de esta prueba es el análisis estático.

1.5.3.2.1 Técnicas de Prueba de caja blanca

Existen varias técnicas de prueba de Caja Blanca, la prueba del camino básico, La prueba de condición, La prueba de flujo de datos y La prueba de bucles.

La prueba del camino básico: Esta prueba permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. Para aplicar esta técnica se deben seguir los siguientes pasos (8):

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

La prueba de condición: Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

La prueba de flujo de datos: Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

La prueba de bucles: Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

1.6 METODOLOGÍAS VS PRUEBAS

Desarrollar un software con calidad depende de un sinnúmero de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo en un determinado proyecto, es trascendental, para el éxito del producto. Las metodologías de desarrollo de software abarcan todo el ciclo de vida del software, y se definen como "un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software" (13).

En los últimos años se han desarrollado dos corrientes en lo referente a los procesos o metodologías de desarrollo de software, los llamados métodos tradicionales o pesados y los métodos ligeros o ágiles. La diferencia fundamental entre ambos radica en que mientras los métodos tradicionales como por ejemplo la metodología RUP, intentan conseguir el objetivo común por medio de orden y documentación. Por otra parte los métodos ligeros como la metodología XP, ponen vital importancia en la capacidad de respuesta a los cambios y al mantener una buena relación con el cliente para llevar al éxito el proyecto

1.6.1 RATIONAL UNIFIED PROCESS - RUP

En primer lugar el proceso unificado es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo el Proceso Unificado es más que un simple proceso, es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. (11)

RUP es un proceso de desarrollo de software que presenta tres características que lo identifican: está dirigido por casos de usos, centrado en la arquitectura, iterativo e incremental. RUP utiliza el Lenguaje Unificado de Modelado (UML) para la realización de todos los esquemas. El proceso de desarrollo de RUP está dividido en fases y flujos de trabajo que unidos conforman el ciclo de vida del proyecto.

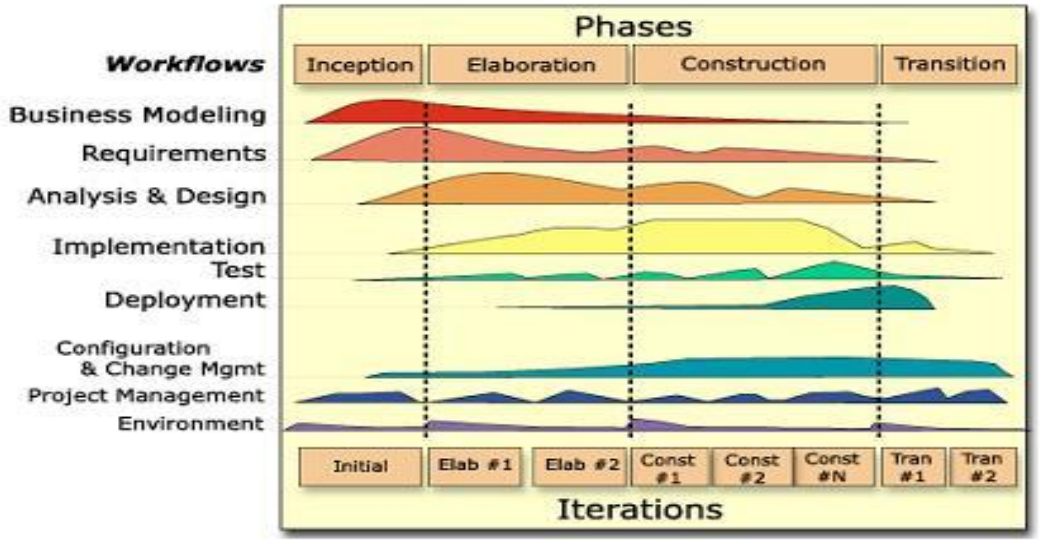

Fig 1.1 Rational Unified Process

1.6.1.1 FASES

1. **Inicio:** El Objetivo en esta etapa es determinar la visión del proyecto.
2. **Elaboración:** En esta etapa el objetivo es determinar la arquitectura óptima.
3. **Construcción:** En esta etapa el objetivo es obtener la capacidad operacional inicial.
4. **Transición:** El objetivo es obtener el release del proyecto.

1.6.1.2 FLUJOS DE TRABAJOS PRINCIPALES

1. **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
2. **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
3. **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
4. **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
5. **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida del software.

6. **Instalación o despliegue:** Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
7. **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
8. **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
9. **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Esta investigación se centra principalmente en el flujo de trabajo de prueba.

RUP plantea que los principales objetivos de las pruebas son:

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas del sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias solo al final de la iteración.
- Diseñar e implementar las pruebas creando los casos de prueba que especifican que probar, creando los procedimientos de prueba que especifican como realizar las pruebas y creando, si es posible, componentes de prueba ejecutables para automatizar las pruebas.
- Realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo como diseño o implementación, de forma que los defectos importantes puedan ser arreglados

Para la realización de estos objetivos RUP define una serie de artefactos y trabajadores, entre los trabajadores que se proponen se encuentran, diseñador de pruebas, Ingeniero de componentes, Ingeniero de pruebas de integración, ingeniero de pruebas de sistema, así mismo estos especialistas desarrollan una serie de artefactos propuestos por esta metodología para las actividades involucradas en el proceso de prueba.

Un **artefacto** es un fragmento de información que es producido, modificado o usado durante el proceso de desarrollo de software. No siempre se crean los mismos artefactos, esto

dependerá de los requisitos que plantee el cliente y de las características del proyecto. Para la fase de pruebas se definen los siguientes (11):

1.6.1.3 DESCRIPCIÓN DE LOS ARTEFACTOS

1.6.1.3.1 Modelo de Pruebas

El artefacto modelo de prueba describe principalmente cómo se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema. Puede describir también cómo se han probado aspectos específicos del sistema, por ejemplo, si la interfaz de usuario del sistema cumple con su objetivo y describe el cumplimiento de los requisitos funcionales y no funcionales del sistema. Es una colección de casos de pruebas, procedimientos de prueba y componentes de prueba.

1.6.1.3.2 Caso de prueba

El diseño de casos de prueba es uno de los pasos más importantes al realizar una estrategia de pruebas, ya que estos constituyen la fuente para evaluar los resultados del software. Especifican una forma de probar el sistema, incluyendo la entrada o resultado con que se va a probar y las condiciones bajo las que ha de probarse.

Una de las principales características que deben presentar los casos de prueba es su carácter abarcador, es decir, deben ser completos y estar adaptados al producto, pues deben ofrecernos la posibilidad de encontrar la mayor cantidad de errores posibles en un tiempo y costo no muy elevados.

1.6.1.3.3 Procedimiento de Prueba

Un procedimiento de prueba especifica cómo realizar uno o varios casos de pruebas ó partes de éstos. Un procedimiento de prueba puede ser una instrucción para un individuo sobre cómo realizar un caso de prueba manualmente, ó una especificación de cómo interactuar manualmente con una herramienta de automatización de pruebas, para crear componentes ejecutables de prueba.

1.6.1.3.4 Componente de Prueba

Un componente de prueba automatiza uno o varios procedimientos de prueba o partes de ellos. Estos pueden ser desarrollados utilizando un lenguaje de guiones o un lenguaje de programación, o pueden ser gravados con una herramienta de automatización de pruebas. Los componentes de pruebas se utilizan también para probar los componentes en el modelo de implementación, proporcionando entradas de pruebas, controlando y motorizando la ejecución de los componentes a probar e informando de los resultados de las pruebas. Los componentes de pruebas pueden ser implementados usando tecnología de objetos.

1.6.1.3.5 Plan de prueba

La construcción de un buen Plan de Pruebas es el principal factor de éxito para la puesta en práctica de una estrategia de pruebas que permita entregar un software de mejor nivel. Es el artefacto que permite trazar el tipo de prueba que se le va a aplicar al producto, cuyo propósito es dejar de forma explícita el alcance, el enfoque, los recursos requeridos, el calendario y los responsables del proceso de pruebas. Durante el desarrollo del software se diseña el plan de prueba con el objetivo de asegurar que todos los requisitos tanto funcionales como de rendimiento, se satisfagan.

1.6.1.3.6 Defecto

Un defecto es un síntoma de un fallo en el software o un de un problema descubierto en una revisión, el cual puede ser utilizado para localizar cualquier cosa que los desarrolladores necesiten registrar cómo síntoma de problema en el sistema.

1.6.1.3.7 Evaluación de Prueba

Es una evaluación de los resultados de los esfuerzos de prueba, tales como la cobertura del caso de prueba, de código y el estado de los defectos. La evaluación es realizada por los diseñadores quienes comparan los resultados obtenidos con los objetivos trazados en el plan de prueba. Durante la evaluación de las pruebas, se realizan métricas que permiten determinar el nivel de calidad del software y qué cantidad de pruebas se deben realizar.

Luego de expuestas la principales características de la metodología RUP, se puede concluir que la misma tiene como objetivos, asegurar la producción de un software de calidad dentro de plazos y presupuestos predecibles, por lo que requiere de cierta preparación así como de disponer de recursos para aplicarla, en el tiempo y costo predefinidos dentro del proyecto. También se puede asegurar que es una metodología aplicable a proyectos que posean gran

cantidad de personal, tiempo para su desarrollo y un presupuesto que permita cubrir el desarrollo del mismo.

1.6.2 EXTREME PROGRAMMING – XP

La programación extrema es una de las metodologías desarrollo de software ágil mas usadas para proyectos de corto plazo y equipos pequeños. Se le llama extrema, porque define pocas reglas y pocas prácticas a través de la eliminación de procesos y técnicas que carecen de valor. En el caso de los proyectos que trabajan con XP, deben seguir procesos disciplinados, pero más que eso, deben combinar la disciplina con la adaptabilidad necesaria del proceso.

La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

XP, dadas sus características, es la metodología más apropiada para un entorno caracterizado por requerimientos cambiantes originados mayormente por un mercado inestable y el inminente avance de la tecnología y los negocios, por tanto es recomendable aplicarla en caso de que los requisitos estén vagamente definidos y los clientes se pueden involucrar la mayor parte del tiempo en el desarrollo del proyecto.

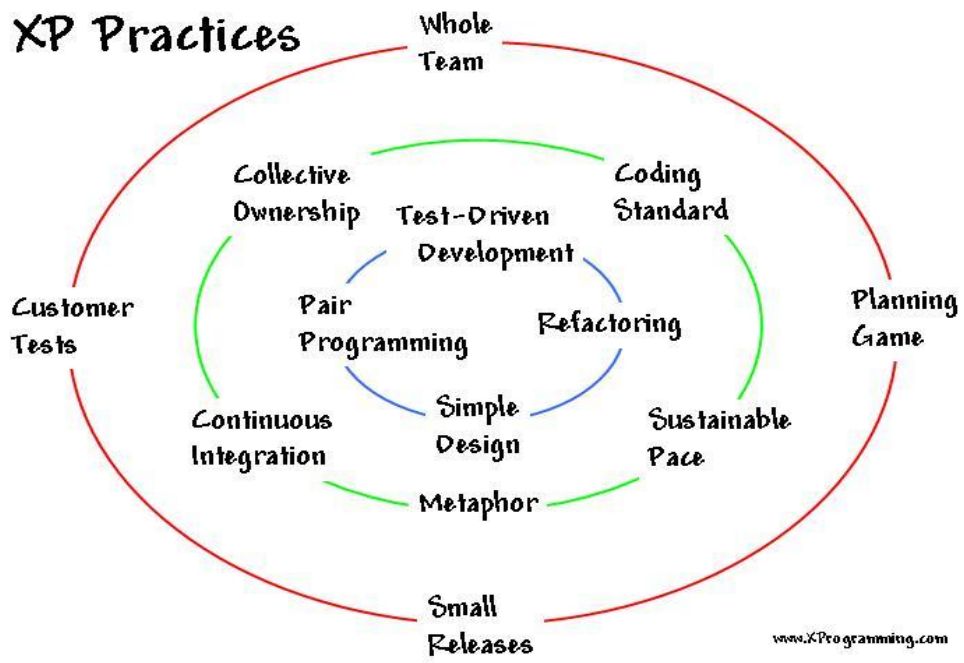

Fig1.2 eXtreme Programming

1.6.2.1 TIPOS DE PRUEBAS

Esta metodología se basa fundamentalmente en (14):

- **Pruebas Unitarias:** Se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.
- **Pruebas de aceptación:** El cliente es el responsable de definirlas, no necesariamente de implementarlos. Él es la persona mejor cualificada para decidir cuál es la funcionalidad más valiosa.

XP insiste en la importancia de una aplicación adecuada y eficiente de las pruebas. Se harán pruebas todo el tiempo, no sólo de cada nueva clase (**pruebas unitarias**) sino que también los clientes comprobarán que el proyecto va satisfaciendo los requisitos (**pruebas funcionales**)

Las pruebas de integración se efectuarán siempre, antes de añadir cualquier nueva clase al proyecto, o después de modificar cualquiera existente (**integración continua**).

1.6.2.2 ROLES RELACIONADOS CON LAS PRUEBAS

Programador: Escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.
Cliente/Usuario: El cliente escribe las UserStories (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. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.
Encargado de pruebas: El 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.

En XP aumentar la calidad conduce a que el proyecto pueda realizarse en menos tiempo. En efecto, en cuanto el equipo de desarrollo se habitúe a realizar pruebas intensivas y se sigan estándares de codificación, poco a poco comenzará a avanzar mucho más rápido de lo que lo hacía antes, mientras la calidad del proyecto se mantiene asegurada por las pruebas, lo que

conlleva mayor confianza en el código y, por tanto, mayor facilidad para adaptarse al cambio, sin estrés, lo que hace que se programe más rápido. Las técnicas de eXtreme Programming proporcionan un camino para obtener productos de calidad medible en el desarrollo de proyectos de software libre. El hecho de que sea una metodología ligera, la hace especialmente idónea para un entorno heterogéneo.

1.7 LOS VIDEOJUEGOS Y LA REALIDAD VIRTUAL

El mundo de hoy esta en constante desarrollo, y con el se desarrolla también el modo de percepción que tiene el hombre con respecto a él. Muy particularmente, los juegos han contribuido a desarrollar ciertas habilidades, y no solo en el campo físico, sino que también en el cognitivo. Del mismo modo, el hombre ha desarrollado diferentes tipos de juegos, tales como los tradicionales, los juegos de computadoras, los videojuegos y los juegos de Realidad Virtual, los cuales se desprenden de los dos últimos mencionados.

Un videojuego es un programa informático, creado expresamente para divertir, basado en la interacción entre una persona y un aparato.

Generalmente para dicha interacción es necesaria la intervención de un dispositivo conectado a una consola, en la cual se generan las imágenes animadas que aparecen en la pantalla del monitor. Algunos videojuegos han sido utilizados con fines educativos, y les han servido a muchas instituciones, para explorar las posibilidades de aprendizaje y el potencial del alumnado, además para motivar al alumno en el aprendizaje y facilitar la comprensión de los mismos a través de la inmersión en el juego.

Estos juegos tienen como inconveniente, que suelen abstraer de la realidad a muchos jugadores, los cuales son en su gran mayoría jóvenes.

Por otra parte, los juegos de la computadora se juegan tanto para la diversión como para aflojar las tensiones de estas épocas .Estos juegos no solo son utilizados por los jóvenes y los niños, sino que también son utilizados por adultos, ya sean hombres o mujeres, porque no distinguen ni sexo ni edad. Estos juegos representan un factor de gran impacto en el desarrollo de la coordinación y de la destreza de la mano y el ojo en los jugadores. Debido a su atractivo y a su constante desarrollo, existen algunos casos en los que los jugadores se llegan a enajenar a de la realidad, al igual que sucede en el caso de los videojuegos.

En el caso particular de los juegos de Realidad Virtual se tiene que los mismos constituyen un paso de avance tanto para la ciencia, como para la humanidad. Acerca de la Realidad Virtual, existen posiblemente tantas definiciones como investigadores haya, pues su reciente y rápida evolución no ha permitido establecer una definición clara. Por ese motivo no debe resultar raro que cada persona emita una idea diferente acerca de lo que entienden por Realidad Virtual. A continuación se presentan algunas de estas definiciones:

- Técnica informática que utiliza pantallas montadas en cascos, sensores, anteojos especiales y dispositivos tridimensionales para que el usuario interactúe con el medio simulado (15).
- Un medio ambiente simulado en el cual se puede estar inmerso. Una realidad virtual provee un ambiente artificial convincente a los sentidos (16).
- Concepto que engloba a las tecnologías que simulan imágenes, sonidos y otras sensaciones de forma que el usuario puede percibir objetos, espacios y personajes imaginarios como si existieran realmente (17).
- Tecnología informática y de comunicaciones que sumerge al usuario en un entorno virtual de naturaleza espacio-sensitiva por medios artificiales que le permiten actuar con el sistema de forma interactiva (18).
- Conjunto de tecnologías que reproducen una realidad proyectada por un computador mediante la combinación de hardware y software. En términos rigurosos, la proyección virtual no es tangible, solo analizada por los sentidos visuales y auditivos respondiendo a los estímulos y sensaciones (19).

La realidad virtual es una técnica actual que permite adentrar al usuario que la utiliza, en un entorno digital creado por el propio hombre. En este entorno se pueden percibir distintas sensaciones y sonidos, basados en estímulos a los órganos sensoriales, e interactuar con imágenes tridimensionales y dinámicas con alto contenido gráfico, y estos elementos en su conjunto, proporcionan un impresionante sentido de la realidad.

La realidad virtual puede ser de dos tipos: inmersiva y no inmersiva. Los métodos inmersivos de realidad virtual con frecuencia se ligan a un ambiente tridimensional creado por un ordenador, el cual se manipula a través de cascos, guantes u otros dispositivos que capturan la posición y rotación de diferentes partes del cuerpo humano. La realidad virtual no inmersiva también utiliza el ordenador y se vale de medios como el que actualmente nos ofrece Internet, en el cual podemos interactuar en tiempo real con diferentes personas en espacios y

ambientes que en realidad no existen sin la necesidad de dispositivos adicionales al ordenador (20)

Los juegos de Realidad Virtual que se desarrollan en la UCI, son generalmente no inmersivos y poseen de forma general características especiales. Se debe contar con conocimientos avanzados en programación como C++, Phyton, OpenGL, entre otros. El programador o el equipo de programadores deben estar capacitados para proveer al juego de efectos naturales, luces y sombras, rendimiento y detalles prácticos, fluidos en tiempo real, entre otros aspectos para enriquecer la calidad de juego.

1.8 TIPOS DE PRUEBAS QUE SE LE REALIZAN A LOS JUEGOS

En la mayoría de los juegos que se desarrollan a nivel mundial, no existe una estrategia de pruebas definida para probar a este tipo de software, generalmente las pruebas que se les desarrollan a estos juegos, son pruebas que realizan los mismos programadores al código, en el cual introducen llamadas a métodos que no existen o que aun no están implementados, la idea es implementar los métodos en la clase a prueba hasta que fallen y luego implementar los métodos de uno en uno hasta satisfacer los requisitos del método y luego pasan al siguiente método. Este tipo de prueba les permite probar como están implementadas las distintas funcionalidades que permiten que el juego funcione correctamente.

También se le hacen pruebas a un entregable (release) del propio juego, entregándolo a una parte del público, de manera que estos jugadores le encuentren problemas y comuniquen los mismos a la entidad que desarrolla el juego (21).

1.9 ESTRATEGIAS DE PRUEBAS

El desarrollo de sistemas de software implica una serie de actividades de producción en las que las posibilidades de que aparezca un fallo humano son enormes. Los errores pueden empezar a darse desde el primer momento del proceso, en el que los objetivos pueden estar especificados de forma errónea o imperfecta así como en posteriores pasos de diseño y desarrollo. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo de software ha de ir acompañado de una actividad que garantice la calidades por ello que muchos son los especialistas que plantean que el proceso de pruebas debe ser llevado a cabo desde el inicio del ciclo de vida del software, si se quiere obtener un producto de alta calidad , igualmente, para la correcta realización de las pruebas es imprescindible llevar a cabo una buena estrategia de pruebas, dicha estrategia debe abarcar planificación,

el diseño de casos de prueba, la ejecución y los resultados, también se deben tener en cuenta los recursos que se van a emplear tanto humanos como materiales. (8)

La planificación de la **Estrategia de Prueba** puede reducir significativamente el esfuerzo necesario para el desarrollo de las pruebas adecuadas, reducir el tiempo de realización y ejecución de las mismas y disminuir los altos costos que se generan.

CAPITULO 2 SITUACIÓN ACTUAL

INTRODUCCIÓN

El estudio del panorama actual es un factor de vital importancia durante el desarrollo de toda investigación científica. Un buen estudio de la situación actual con respecto al tema de la investigación realizada, brinda la posibilidad de estar al tanto de lo que acontece en el mismo campo en el que se desarrolla dicha investigación, posibilita además, conocer la competencia, y los diferentes proyectos de investigación que se desarrollan. Para este caso especifico el estudio será llevado a cabo en La Universidad de las Ciencias Informáticas (UCI)la cual se encuentra dividida en 13 facultades, 10 en la sede central y 3 facultades regionales, en cada una de ella se desarrollan diferentes polos productivos.

Particularmente la facultad 5 se encuentra dividida en dos polos los cuales son Polo de Hardware y Automática y Polo de realidad virtual. Este ultimo abarca una serie de líneas de trabajo entre las que se encuentran:

1. Investigación
2. Desarrollo de Soluciones
3. Soporte
4. Tecnología
5. Capacitación

Dentro de Desarrollo de Soluciones Las líneas potenciadas actualmente son:

- Video-Juegos de Entretenimiento
- Simuladores para Entrenamiento – Evaluación.
- Maquetas Virtuales.
- Laboratorios Virtuales.
- Diseño y Realización de Entornos y Animaciones 3D.
- Aplicaciones de Realidad Virtual para la salud
- Evaluadores teóricos basados en situaciones 3D.
- Herramientas para el desarrollo de Software para Realidad Virtual (SCeneToolKit).
- Juegos

El posterior desarrollo de esta investigación estará vinculado con las líneas de desarrollo Video-Juegos de Entretenimiento y Juegos.

La facultad 5 cuenta con nueve proyectos de Realidad Virtual, de los cuales tres están relacionados con las líneas de desarrollo antes mencionadas, ellos son:

- Compilación de juegos.
- Juegos CNEURO.
- Juegos de Consola.

Con el objetivo de conocer la situación actual por la que atraviesan estos proyectos, referente a las estrategias de pruebas que se ponen en marcha por parte de los mismos, se lleva a cabo una entrevista a los Líderes de cada uno de ellos, la cual arrojo los siguientes resultados:

2.1 RESULTADOS DE LAS ENTREVISTAS

2.1.1 JUEGOS CNEURO

La entrevista fue aplicada al ingeniero Yulier Casas Estrada el cual se encuentra desempeñando el rol de jefe de proyecto. El producto que se desarrolla es un juego para la intervención de la Discalculia en escolares de 6to grado. Se espera que con este producto mejore la afectación que presentan estos niños, además de que los que no la padecen pueden utilizarlo como material de estudio. En un futuro se piensa desarrollar un juego para cada uno de los grados de escolaridad primaria, desde 1ero al 5to.El producto consta de tres módulos:

- Kernel
- Algoritmo adaptativo
- Niveles

Actualmente se está trabajando sobre el producto Venganza en su versión 1.0, para el se espera tener una primera entrega con tres niveles del juego terminado en septiembre del 2008.

El proyecto esta compuesto por 16 estudiantes y 6 profesores, en el mismo no existe un equipo de calidad, el jefe de proyecto y un estudiante son los responsables de llevar a cabo

todas las actividades relacionadas con la calidad de este aunque actualmente se esta desarrollando una tesis encaminada a asegurar la calidad del mismo.

CNeuro se esta desarrollando siguiendo la metodología RUP, la cual es una metodología de desarrollo tradicional, pero se plantea la necesidad del uso de una metodología mas ágil debido a las características que encierran los juegos de este tipo. Todavía no han llevado a cabo revisiones al guión y a los requerimientos, ni se han realizado pruebas, pero se tiene conocimiento de que existe una estrategia para las mismas y que serán aplicadas a medida que se liberen entregables o por iteraciones, dicha estrategia esta siendo trazada por parte de dos estudiantes del grupo de calidad de la facultad. El líder del proyecto plantea que a su consideración las pruebas más importantes que se realizan dentro del proyecto son las pruebas que se realizan a nivel de unidad y a nivel de sistema, y le atribuye cierta importancia a las pruebas que llevan a cabo los clientes.

El ingeniero Yulier Casas opina que una de las características que hacen especiales a los juegos de Realidad Virtual es que se desarrollan a través de guiones y a partir de los mismos se elaboran los casos de uso y el levantamiento de requisitos del mismo. Estos guiones son entregados inicialmente por el cliente pero luego el equipo de desarrollo se reúne con este para redefinir el mismo, en este proyecto actualmente no se posee ningún documento oficial que establezca como realizar un guión de hecho este proyecto es uno de los primeros en usarlo en la Universidad.

2.1.2 JUEGOS DE CONSOLA

Juegos de Consola surge ante la necesidad del Consejo de Estado de modernizar los parques de diversiones con la introducción de video-juegos capaces de proporcionar una sana recreación a los niños principalmente en el rango de 8 a 12 años de edad, el presente se limita a definir, diseñar e implementar un entorno gráfico 3D en tiempo real con sonido realista e interacción con el usuario para lograr la integración de un juego de conducción de automóviles que será instalado en el Parque Mariposa. El mismo consta de ocho módulos:

- ASE
- Detención de colisiones
- Controladores (Joystick)
- Sonido (Avi, Mp3)
- Lógica del juego

- Modelamiento física-matemático
- Menú
- RPM

En el proyecto actualmente se encuentran trabajando 22 estudiantes y 2 profesores para desarrollar el producto Rápido y Curioso el cual se espera terminar el 1ro de julio del 2008.

Con el objetivo de conocer la situación actual de este proyecto se le aplicó una entrevista al ingeniero Yoander Cabrera Díaz el cual se encuentra desempeñando el rol de jefe de proyecto.

El líder de este proyecto plantea que se tiene conocimiento de las Metodologías de software existentes, entra ellas citó RUP y XP, plantea además que en su proyecto se trabaja siguiendo la Metodología RUP con algunas adaptaciones necesarias para este tipo de proyecto.

Se tienen conocimiento de algunas de las pruebas que propone RUP, Caja negra y Caja Blanca, en juegos consola no hay definida una estrategia de pruebas, simplemente se diseñan los casos de prueba, se implementan y posteriormente se documentan los resultados arrojados. Las pruebas que se aplican son:

- Validación
- Sistema
- Aceptación

Estas pruebas son, a consideración del jefe de proyecto, las mas importantes y se llevan a cabo cada vez que se termina un entregable aunque el plantea que tiene conocimiento de que es necesario comenzarlas desde el inicio del proyecto. En el proyecto no se han llevado a cabo revisiones a los requerimientos.

El equipo de calidad esta integrado por dos estudiantes y un profesor al frente del mismo. En el proyecto no se realiza la planificación de las pruebas, el diseñador es el encargado de diseñarlas, el probador de ejecutarlas y el profesor que esta al frente del equipo las documenta.

El ingeniero Yoander Cabrera Díaz opina que el software de juegos difiere mucho de un software de gestión, debido a que estos llevan Inteligencia artificial, Gráficos por Computadoras y el uso de guiones.

2.1.3 COMPILACIÓN DE JUEGOS

Con el objetivo de conocer la situación en la que se encuentra actualmente el proyecto Compilación de Juegos se le aplicó una entrevista al ingeniero Dagoberto Marrero López el cual se encuentra desempeñando el rol de jefe de proyecto.

Compilación de Juegos es un proyecto internacional que consiste en la realización de dos juegos destinados a la población infantil de Venezuela, entre las edades de 9 a 11 años, son juegos en dos dimensiones con el objetivo de que los niños venezolanos sepan la importancia de la revolución energética que se lleva a cabo en su país y de como influye esta en el cuidado de nuestro planeta. Este proyecto comenzó el 14 de febrero 2008, esta compuesto por 18 estudiantes y 4 profesores los cuales se encuentran desarrollando dos juegos:

- Juego Revolución Energética.
- Juego del Petróleo.

Se espera que para el 20 de septiembre del presente año se liberen estos productos.

En el proyecto se tiene conocimiento de la existencia de algunas Metodologías como RUP y XP y se trabaja guiándose por la metodología RUP, pero con adaptaciones de la misma, puesto que se plantea que RUP posee muchos artefactos y este tipo de proyecto requiere agilidad, lo que inicialmente llevo a sus desarrolladores a pensar que se podía utilizar XP pero esta ultima representa un problema para un software tan complejo que requiere de un diseño riguroso.

Compilación de Juegos es un proyecto que comenzó hace apenas tres meses por lo que todavía no se ha concretado una estrategia de pruebas pero se han llevado a cabo revisiones al guión técnico. No se han aplicado pruebas, pero se le atribuye gran importancia a cada una de ellas y con respecto a esto el jefe de proyecto plantea “aun no se le han realizado pruebas al software, pero se espera porque el proyecto es internacional y los clientes (Venezuela) van a exigir calidad”.

Se piensa que la aplicación de las pruebas se realice por módulos y que sean ejecutadas por parte del equipo de calidad del proyecto, el cual esta integrado por un estudiante y un profesor al frente, asesor de calidad.

El ingeniero Dagoberto Marrero López opina que una de las características que hacen diferente al software de juegos del de gestión es el uso de guiones y le atribuye gran importancia a los mismos, debido a que el desarrollo del juego parte desde aquí. Para la realización del proyecto Compilación de Juegos se elaboraron dos guiones, uno de contenido y otro técnico.

2.2 RESULTADOS GENERALES DE LOS PROYECTOS

Después de realizadas y procesadas las entrevistas aplicadas a los jefes de proyectos podemos afirmar a modo de conclusión que dentro de los elementos mas significativos que fueron detectados esta el desconocimiento por parte de los líderes de proyecto de algunas de las metodologías, sin embargo todos están utilizando RUP y señalan que existen problemas a la hora de desarrollar un software de este tipo guiándose por una metodología que genera tantos artefactos. El nivel de conocimiento acerca de las metodologías de software aplicables, se encuentra reflejado en la figura 2.1.

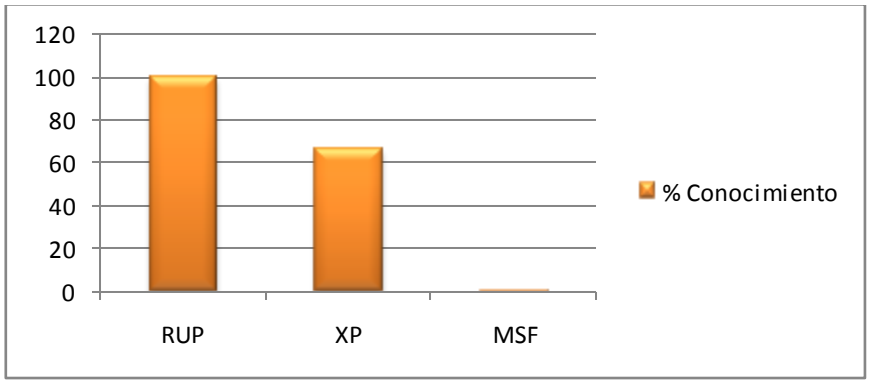

Fig. 2.1 Conocimiento de las metodologías.

De forma general existe un vago conocimiento de las pruebas que propone la metodología RUP y no está definida una estrategia de pruebas en estos proyectos, con la excepción de CNeuro el cual posee una estrategia de aseguramiento de la calidad que esta siendo trazada por parte de dos estudiantes del grupo de calidad de la facultad.

En uno de estos proyectos de llevaron a acabo pruebas, pero sin el empleo de una estrategia, es decir, al término de un entregable se diseñaron casos de pruebas y se ejecutaron sin

previa planificación. En el caso de los demás proyectos, aun no se han realizado pruebas. De manera general los líderes de proyecto no le atribuyen a las pruebas la misma importancia, el 100% de los lideres considera que las pruebas más importantes son las pruebas que se realizan a nivel de sistema, en segundo plano se encuentran las pruebas que se realizan a nivel de unidad y de aceptación, y por ultimo las pruebas de integración. Este comportamiento se expone en la figura 2.2.

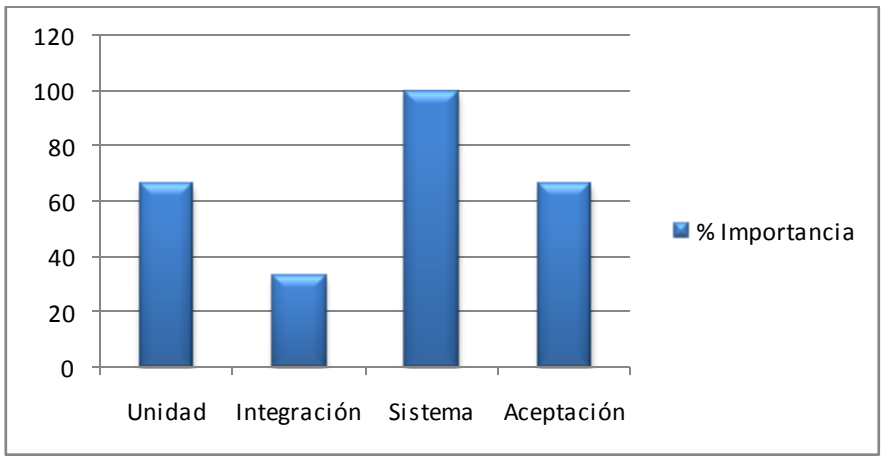

Fig. 2.2 Importancia de las pruebas.

En la siguiente figura se ilustra la importancia que se le atribuye a cada uno de los niveles de pruebas, en cada uno de los proyectos estudiados.(Fig. 2.3)

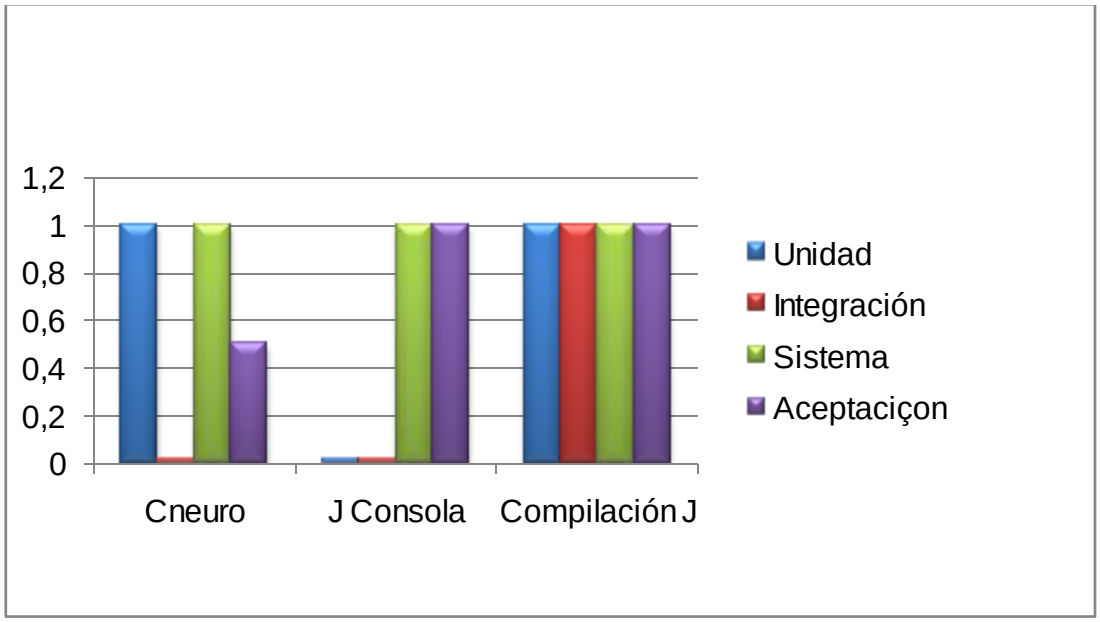

Fig. 2.3 Importancia de las pruebas por proyectos.

En el 100% de los casos, los líderes de proyectos plantean que las pruebas deben comenzar a realizarlas cada vez que obtienen un entregable o un módulo y que se deben llevar a cabo revisiones a los requerimiento dentro del proyecto. Uno de los problemas más críticos que presentan los proyectos de juegos de Realidad Virtual de la facultad 5 radica en la falta de personal que tienen para conformar el equipo de calidad, en la totalidad de los casos el número máximo de estudiantes que posee es dos y no desempeñan un rol determinado sino que se encargan de todas las actividades relacionadas con la calidad en su proyecto.

El uso de guiones es una de las características que diferencian al software de juego de Realidad Virtual del software de gestión y que los jefes de proyecto identifican como la más importante debido a que a partir de aquí es que comienza a elaborarse el juego, en la facultad 5 no existía precedentes del uso de estos guiones, por lo tanto en los proyectos no se cuenta con una guía para redactarlos y mucho menos para revisarlos.

CAPITULO 3: PROPUESTA DE LA SOLUCIÓN

INTRODUCCIÓN

El objetivo de este capítulo es proponer una estrategia de pruebas de forma genérica que permita mejorar el proceso de pruebas a los juegos de Realidad Virtual, se van a proponer las pruebas de software que son más factibles para la utilización en los proyectos de Realidad Virtual de la Facultad 5.

En la actualidad no existe dentro de los proyectos de juegos de Realidad Virtual, una estrategia que permita aplicar las pruebas a software de este tipo, ya que no existían precedentes de los mismos en la universidad. Se hacia necesaria la creación de un grupo de calidad dentro del proyecto, el cual llevara a cabo las actividades de aseguramiento y control de la calidad, dentro de las que entra al realización de las pruebas.

En la Facultad 5 existe un grupo de Aseguramiento de la calidad el cual se encarga de realizarle pruebas a los productos una vez desarrollados en su totalidad. No existe una estrategia de pruebas definida para ser aplicada dentro del proyecto durante todo el desarrollo del ciclo de vida del software, por lo que se proponen una serie de pruebas para ser aplicadas por el quipo de pruebas interno del proyecto, para que luego el equipo de calidad de la facultad reciba un software que contenga la menor cantidad de No Conformidades posibles.

Teniendo en cuenta que un factor indispensable para lograr la calidad del juego, lo constituye la calidad de la historia, la calidad del guión, y la calidad del diseño, la estrategia propuesta abarca listas de chequeos para el guión técnico y para los requerimientos que se derivan del mismo. A continuación se ofrecen las distintas actividades por las cuales esta compuesta la estrategia de pruebas. En esta sección se presenta además un flujo de trabajo que describe las distintas actividades que se deben desarrollar para la realización de las pruebas.

3.1 DEFINICIÓN DE LA ESTRATEGIA DE PRUEBAS A APLICAR.

De acuerdo con la investigación realizada se determinó que por las características que presentan los juegos de Realidad Virtual, la estrategia de pruebas que se propone consta de listas de chequeo además de la aplicación de 4 niveles de pruebas. Esta estrategia está definida incrementalmente de “adentro” hacia “afuera”, o sea a partir de la prueba de unidad establecida para evaluar el funcionamiento de los módulos, la prueba de integración, la prueba del sistema y por último la prueba de aceptación.

Fig.3.1 Actividades propuestas

Listas de chequeo:

- ✓ Revisión al Guión Técnico: Aplicar lista de Chequeo propuesta.
- ✓ Revisión a los Requerimientos: Aplicar lista de Chequeo propuesta.

Niveles de Prueba:

- ✓ Pruebas de Unidad: Probar cada módulo por separado.
- ✓ Pruebas de Integración: Probar la interacción de los módulos.
- ✓ Pruebas de Sistema: Verificar que el programa cumple con los requisitos.
- ✓ Pruebas de Aceptación: Validar el producto con el cliente.

En cada nivel las pruebas se realizarán con el método de caja negra utilizando la técnica Partición Equivalente. Para el nivel de unidad se aplicará además el método de Caja Blanca.

Las pruebas a aplicar por niveles son:

✓ Pruebas de Unidad

- Prueba de Funcionalidad
- Pruebas de Caja blanca

- ✓ Pruebas de Integración
  - Prueba de Funcionalidad
- ✓ Pruebas de Sistema
  - Prueba de Funcionalidad
  - Prueba de Rendimiento
  - Prueba de Resistencia
  - Pruebas de Interfaz de usuario
- ✓ Pruebas de Aceptación

De forma general las pruebas seguirán el curso propuesto en el siguiente flujo de trabajo:

3.2 FLUJO DE TRABAJO

El flujo de trabajo que a continuación se muestra describe las distintas actividades que se llevarán a cabo para realizar cada una las pruebas. De manera general, se debe comenzar con la planificación, en esta actividad se genera el plan de pruebas un artefacto de vital importancia para el desarrollo de las mismas puesto que en este se tienen en cuenta aspectos como los recursos, requerimientos y los resultados que arrojan las pruebas. Posteriormente se desarrollan de forma paralela el diseño y la configuración del ambiente de pruebas. Luego de concluidas estas actividades, se procede a ejecutar las mismas, donde se documentan las no conformidades detectadas y se registran los problemas surgidos durante la jornada de trabajo. Finaliza el flujo con la conclusión de las pruebas, actividad en la que se concilian las no conformidades detectadas con el equipo de desarrollo y se emite el expediente de pruebas.

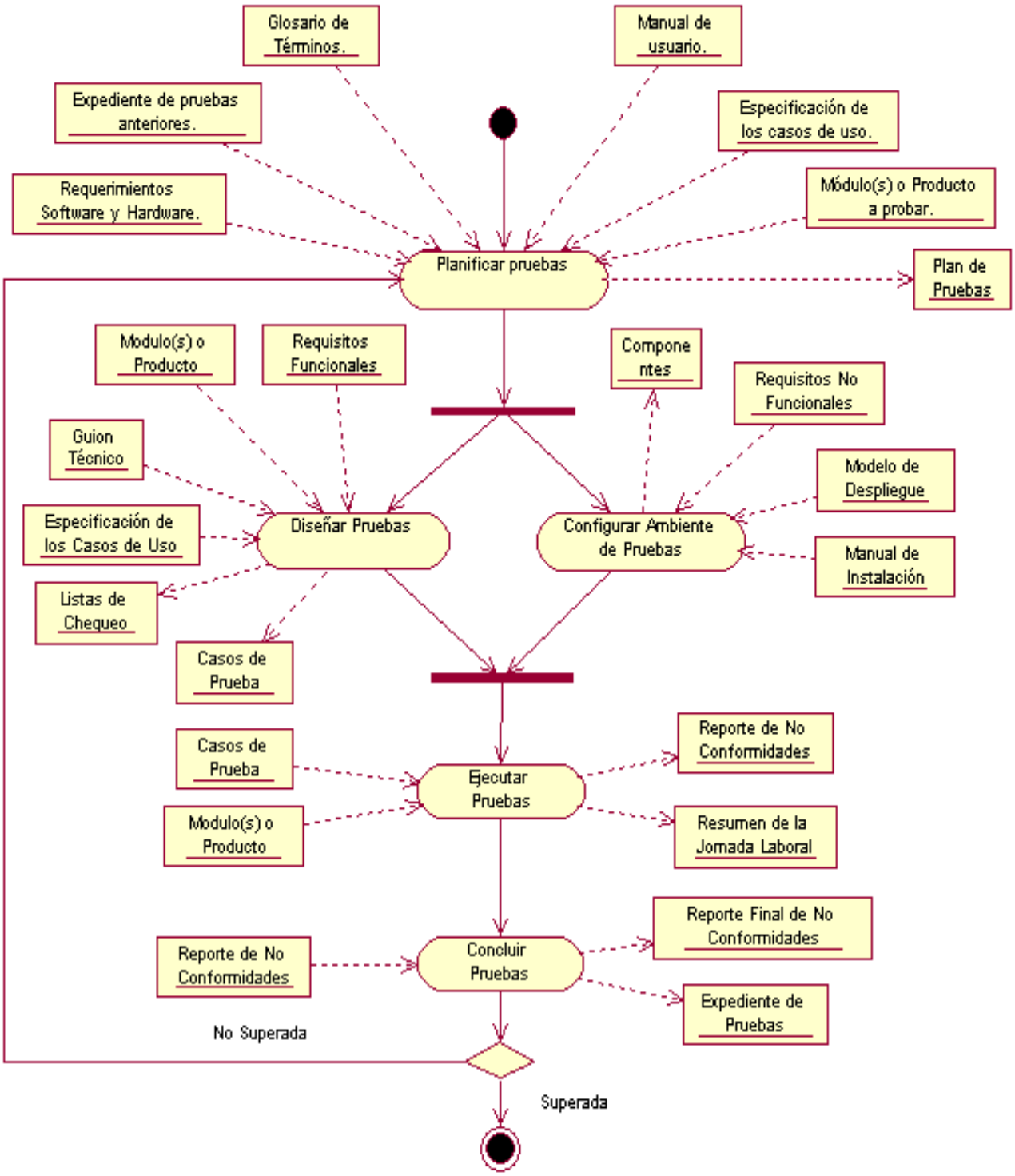

Fig. 3.2 Flujo de trabajo de pruebas

3.2.1 DESCRIPCIÓN DE LAS ACTIVIDADES DEL FLUJO DE TRABAJO

3.2.1.1 Planificar pruebas

El propósito de esta actividad es definir el alcance que van a tener las pruebas y sus objetivos teniendo en cuenta los recursos necesarios para su realización. Cuando se habla de recursos se deben tener en cuenta tanto los materiales como los humanos.

El plan de pruebas puede estar sujeto a cambios a lo largo de toda la vida del proyecto, por tanto, dentro de esta actividad estará contemplada la redefinición del mismo. El administrador de pruebas es el encargado de realizar la actividad de planificación.

Pasos	Entradas	Salidas
1. Identificar los recursos necesarios para realizar las pruebas. • Recursos Humanos. • Recursos Software. • Recursos Hardware. 2. Confeccionar Plan de Pruebas. 3. Aprobación del Plan de pruebas por parte del equipo de desarrollo y el equipo de calidad.	Expediente de pruebas anteriores. Especificación de los casos de uso. Requerimientos Software y Hardware. Módulo(s) o Producto a probar. Manual de usuario. Glosario de Términos.	Plan de pruebas.

3.2.1.2 Diseñar pruebas

El objetivo de esta actividad es identificar, describir y diseñar los casos de prueba, el responsable de realizarla es el diseñador de pruebas. Luego de identificadas las pruebas, el diseñador procede a seleccionar las técnicas que se deben utilizar para el tipo de software con el que se trabaja, analiza los objetivos de las pruebas que se han planificado y desarrolla los casos de prueba.

Pasos	Entradas	Salidas
1. Diseñar los casos de pruebas, determinando las condiciones del juego en el software a probar. 2. Aprobación de los casos de pruebas a aplicar, por parte del equipo de aseguramiento de la calidad.	Requerimientos Software y Hardware. Plan de Prueba. Especificación de Casos de Uso. Guión Técnico. Módulo(s) o producto a probar. Glosario de Términos.	Expediente de pruebas. Para esta actividad debe contener: - Casos de prueba. - Especificación de los casos de prueba.

3.2.1.3 Configurar Ambiente de Pruebas

Esta actividad tiene como propósito asegurar todas las condiciones de hardware y software necesarias para la ejecución de las pruebas, además de garantizar las herramientas necesarias, y la capacitación del equipo de probadores seleccionado previamente para ejecutar las pruebas. Además se incluye la actividad de implementar los casos de prueba que se han definido durante el diseño de las pruebas, automatizando todos los que sean posibles. El encargado de realizar esta actividad es el Ingeniero de componentes, aunque pueden incorporarse el diseñador y el probador.

Pasos	Entradas	Salidas
1. Crear y configurar ambiente de pruebas. 2. Capacitar al equipo de probadores.	Plan de Pruebas. Requerimientos no funcionales. Modelo de Despliegue. Manual de Instalación.	Componentes a automatizar. Entorno de configuración de pruebas.

3.2.1.4 Ejecutar las pruebas

Esta actividad tiene como objetivo llevar a cabo las pruebas de unidad, integración y del sistema del producto desarrollado, así como aplicar las listas de chequeo propuestas. Durante esta actividad se ejecutan las pruebas previamente planificadas y diseñadas. El probador es el encargado de ejecutar todas las pruebas, durante la ejecución de las mismas se realizan reportes de No Conformidades (NC) por parte de cada probador (Ver anexo 4) y se registran los resultados.

Pasos	Entradas	Salidas
1. Ejecutar casos de pruebas y listas de chequeo. 2. Registrar las No Conformidades (NC). 3. Realizar un resumen diario de la jornada de trabajo.	Plan de Pruebas. Casos de Pruebas.	Reporte de NC. Resumen de la Jornada de trabajo.

3.2.1.5 Concluir pruebas

En esta actividad se realiza un análisis final de las pruebas aplicadas y se confeccionan informes finales. Los diseñadores de pruebas llevan a cabo esta actividad revisando y

evaluando los resultados de las pruebas. Luego el administrador de pruebas, se encarga de emitir el documento donde se registran las NC finales (Ver anexo 8).

Pasos	Entradas	Salidas
1. Confeccionar un documento donde se registren las No Conformidades finales.	Reporte de No Conformidades (NC).	Reporte final de NC.

3.3 DESCRIPCIÓN DE LOS ARTEFACTOS

3.3.1 PLAN DE PRUEBAS

El propósito del plan de pruebas es explicitar el alcance, enfoque, recursos requeridos, calendario, responsables, organización y estrategia de las pruebas. Se puede definir un plan de pruebas de forma global y uno para cada tipo de prueba. El mismo se realizara haciendo uso de la plantilla propuesta por la dirección de calidad de la universidad.

Un plan de pruebas incluye:

1. Introducción.
2. Organización del Equipo de Pruebas.
3. Arquitectura técnica.
4. Especificaciones del Software y Hardware.
5. Descripción del Plan de Pruebas.
6. Descripción de los requerimientos.
7. Estrategia de Prueba.
8. Recursos Requeridos.
9. Plan de proyecto.
10. Calendario y Plazos.
11. Definición de los Entregables.
12. Seguimiento y Reporte de Defectos.
13. Aprobación del Plan.
14. Documentación de los Resultados.

3.3.2 EXPEDIENTE DE PRUEBAS.

El propósito de este artefacto es archivar todos los documentos generados durante la etapa de pruebas.

El expediente de pruebas debe incluir:

1. Diseño de Casos de pruebas.
2. Especificación de Casos de pruebas.
3. Reporte de No Conformidades.
4. Resumen de la Jornada de trabajo.
5. Componentes automatizados.

Todos estos documentos se deben redactar siguiendo las plantillas propuestas y se deben recoger en una carpeta con el nombre de expediente de pruebas.

3.3.3 CASOS DE PRUEBA

El propósito de este artefacto es definir un conjunto de condiciones o variables bajo las cuáles se determinará si el requisito de una aplicación es parcial o completamente satisfactorio. Los casos de pruebas se deben diseñar a partir de la plantilla propuesta por la dirección de calidad de la universidad (Ver anexo 5).

3.3.4 ESPECIFICACIÓN DE LOS CASOS DE PRUEBA

Es un documento donde se describe de forma detallada el alcance de cada caso de prueba así como sus flujos principales y alternos. Este documento se debe desarrollar a partir de la plantilla propuesta. (Ver anexo 6).

Según la IEEE Std 829-1998 (IEEE Standard for Software Test Documentation), este documento debe contener:

1. Descripción.
2. Elementos a probar.
3. Condiciones de ejecución.
4. Entradas.
5. Salidas.
6. Configuración de Ambiente de pruebas.

7. Evaluación de la Prueba.

3.3.5 Reporte de No Conformidades (NC)

En esta tabla se deben recoger las NC detectadas por cada uno de los probadores durante la ejecución de cada caso de prueba. El mismo se debe desarrollar siguiendo el formato propuesto por la universidad (Ver anexo 4).

3.3.6 Reporte final de NC

Es un documento en el cual se recogen todas las NC detectadas durante el proceso de pruebas. Este documento se debe desarrollar a partir de la plantilla propuesta (Ver anexo 8).

3.3.7 Resumen de la jornada de trabajo

En este documento se recogen los resultados de la jornada de trabajo y se hace un resumen de las no conformidades detectadas en el día. (Ver anexo 7).

3.4 Descripción de la estrategia de pruebas a aplicar

3.4.1 Revisión al guión técnico:

Uno de los primeros pasos que se realizan durante la concepción de un juego de Realidad Virtual es la confección del guión técnico. El guión técnico, debe ser elaborado por algún especialista en guiones vinculado al proyecto, a partir del guión de contenidos previamente redactado por los clientes. El mismo debe contener de forma explícita, todas las características del juego, ya sean ventanas, menús, opciones, acciones que debe realizar el usuario y las respuestas que debe dar el sistema. La revisión al guión técnico debe ser llevada a cabo por el equipo de aseguramiento de la calidad, pero en este caso debe estar integrado a este equipo un representante del equipo de desarrolladores. Esta actividad debe comenzar de forma paralela al desarrollo del guión, es decir, en la Fase inicial del producto.

Los guiones técnicos constituyen la base para la posterior captura de requisitos, confección de los casos de usos y construcción del juego en general, por tanto, para garantizar que los mismos sean lo más concisos, claros y explicativos posible, se debe hacer una revisión del documento mediante una lista de chequeo para guiones de Realidad Virtual.

Proponer una lista de chequeo de forma genérica no seria factible debido a que cada juego encierra características que lo hacen único, por lo tanto es recomendable que se elabore una lista para cada guión, teniendo en cuenta el formato en el que se describe el juego. De forma general se elaboró una lista de chequeo que puede servir como punto de partida para la revisión y elaboración de los mismos (Ver anexo 2).

3.4.2 REVISIÓN A LOS REQUERIMIENTOS

Según la IEEE 830 algunas de las características que deben tener las especificaciones de requisitos son: correcta, inequívoca, completa, consistente, comprobable, modificable e identificable; otros autores le suman a este conjunto: ordenable, verificable, trazable y no ambigua.

Debido a la importancia que se le confiere al proceso de levantamiento de requerimientos es recomendable que el Equipo de Calidad revise el documento de especificación de requerimientos. La revisión debe comenzar durante la Fase inicial del producto, empleándose para la misma, una lista de chequeo propuesta para esta actividad (ver anexo 3), para su confección se tuvieron en cuenta algunos de los criterios que a continuación se exponen.

- Necesario: Si el sistema puede satisfacer las necesidades reales sin el requisito, entonces no es necesario.
- Factible: El requisito puede ser desarrollado dentro del tiempo y presupuesto.
- Correcto: Los hechos relacionados a los requisitos son precisos, y técnica y legalmente posibles.
- Conciso: El requisito esta escrito de forma simple, no expresa más de una funcionalidad.
- No ambiguo: El requisito puede ser interpretado en solo una forma.
- Completo: El requisito expresa una idea o enunciado en su totalidad y las condiciones bajo las que se aplica el requisito se encuentran declaradas.
- Consistente: No entra en conflicto con otros requisitos.
- Verificable: La implementación del requisito en el sistema puede ser probada.
- Independiente del diseño: El requisito no debe asumir una solución de implementación específica.
- No redundante: No existe duplicación del requisito.
- Asignado identificador único: Cada requisito debe tener un identificador único.
- Uso de frases especulativas: p.e. usualmente, generalmente, típicamente.

- Localización de conceptos condicionales: p.e. quizá, probablemente.
- Uso de un único verbo por requisito: lo que permite comprobar la existencia de una única necesidad en cada requisito
- Uso de términos ambiguos: 'bastante', 'suficiente', 'seguro', 'usable'.
- Uso de frases no completas: 'más adelante', 'en un futuro'.

Para mejor entendimiento de la estrategia propuesta la descripción de cada una de las pruebas constara de los siguientes aspectos:

- Descripción: [Descripción general de la prueba a aplicar].
- Objetivo: [Objetivo que persigue la prueba a aplicar].
- Técnica: [Descripción detallada de la manera en la que se prueba].
- Responsable de ejecutar la prueba [Responsable de ejecutar la prueba].
- Responsable de diseñar la prueba [Responsable de diseñar la prueba].
- Comienzo de la prueba [Momento en que comienza la prueba].
- Procedimiento para realizar la prueba [Breve descripción del proceso].

3.4.3 PRUEBAS DE UNIDAD

3.4.3.1 Pruebas de Funcionalidad

- Descripción: Estas pruebas se aplican para verificar que el módulo se adecua a los requerimientos funcionales y a lo descrito en el guión técnico.
- Objetivo: Ejecutar los métodos de las clases bajo distintas condiciones o distintas variables de entrada y analizar el resultado.
- Técnica: La técnica a emplear será Partición de Equivalencia. Esta técnica consiste en invocar cada clase o método con datos válidos, inválidos y con los valores de los límites para asegurar que los datos fluyen correctamente.
- Responsable de ejecutar la prueba :Probador
- Responsable de diseñar la prueba: Diseñador de Pruebas.
- Comienzo de la prueba: Debe comenzar durante la Fase de elaboración del producto, una vez terminado el primer módulo.
- Procedimiento para realizar la prueba: Inicialmente se deben identificar las clases de equivalencia (clases válidas e inválidas), a partir de estas clases se diseñan los casos de prueba utilizando la plantilla propuesta por la universidad (ver anexo 5).Los casos de prueba deben ser aprobados por el Administrador de pruebas. Antes de comenzar las pruebas se

crea un expediente el cual incluye todo lo que se genere durante el proceso de pruebas. Una vez diseñados y aprobados los casos de prueba el probador los ejecuta y documenta los resultados obtenidos, si se detecta alguna no conformidad, estas deben ser registradas en la tabla de No Conformidades que se encuentra en la plantilla de Diseño de Casos de Prueba (Ver anexo 4).

3.4.3.2 Pruebas de Caja Blanca

- Descripción: Prueba dirigida al código del programa para verificar que se cumpla cada sentencia al menos una vez.
- Objetivo: Evaluar la lógica interna del programa y verificar el comportamiento de un método o función dada una entrada específica.
- Técnica: La técnica a emplear será Camino Mínimo.
- Responsable de ejecutar la prueba :Probador
- Responsable de diseñar la prueba: Para diseñar este tipo de pruebas es de vital importancia que el Diseñador tenga conocimientos de programación, por lo que se sugiere que para este caso especifico además del Diseñador de Pruebas intervenga un programador miembro del equipo de desarrollo.
- Comienzo de la prueba: Debe comenzar durante la Fase de elaboración del producto, inmediatamente se vayan implementando los módulos.
- Procedimiento para realizar la prueba: El jefe de proyecto entrega el código del módulo a probar al diseñador, el cual luego de aplicar la técnica seleccionada, obtiene la cantidad de casos de pruebas que debe diseñar. Luego estos casos de pruebas son ejecutados por el probador.

3.4.4 PRUEBAS DE INTEGRACIÓN

3.4.4.1 Pruebas de Funcionalidad

- Descripción: Esta prueba esta enfocada a verificar el cumplimiento de los requisitos del sistema una vez integrados los módulos.
- Objetivo: Evaluar el comportamiento de un módulo cuando su funcionamiento depende de servicios prestados por otro(s) módulo(s).
- Técnica: La técnica a utilizar para este tipo de pruebas será la de Partición de Equivalencia, para garantizar que los datos de los módulos que interaccionan fluyen correctamente. Se debe integrar de forma descendente, ir combinando los módulos de los niveles superiores moviéndose hacia los inferiores por la estructura del programa.

- Responsable de ejecutar la prueba: Probador.
- Responsable de diseñar la prueba: Diseñador de Pruebas.
- Comienzo de la prueba: Estas pruebas deben comenzar durante la Fase de Construcción del producto una vez que se obtengan dos o más módulos que estén relacionados.
- Procedimiento para realizar la prueba: Para diseñar los casos de pruebas el diseñador debe identificar las clases de equivalencia, luego de tener las clases válidas e inválidas definidas, comienza con el diseño de los mismos. Los casos de prueba de integración deben ser elaborados a partir de las especificaciones de los casos de uso y deben ser aprobados por el Administrador de pruebas. Luego el probador los ejecuta y documenta los resultados obtenidos, si se detecta alguna no conformidad, estas deben ser registradas en la tabla de No Conformidades que se encuentra en la plantilla de Diseño de Casos de Prueba propuesta por la universidad. (Ver anexo 4)

El primer paso es identificar, por parte del administrador de las pruebas, los módulos críticos, los cuales serán probados lo antes posible. Un módulo crítico es aquel que tiene una o más de las siguientes características: (1) está dirigido a varios requisitos del software; (2) tiene un mayor nivel de control (está relativamente alto en la estructura del programa); (3) es complejo o propenso a errores o (4) tiene unos requisitos de rendimiento muy definidos. (8)

Antes de comenzar las pruebas se crea un expediente el cual incluye todo lo que se genere durante el proceso de pruebas. Los errores encontrados deben ser registrados en la tabla de No Conformidades que se encuentra en la plantilla de Diseño de Casos de Prueba propuesta por la universidad. (Ver anexo 5)**.** Para la corrección de estos errores se debe informar diariamente al equipo de desarrollo, quienes informan al equipo de pruebas los cambios realizados.

Es aconsejable que paralelamente al proceso de integración se apliquen las pruebas de regresión, con el objetivo de asegurar que las fallas detectadas en los módulos sean corregidas y que no se introduzcan nuevos errores al tratar de solucionar los detectados anteriormente. Las pruebas de regresión consisten en aplicar el mismo proceso de pruebas planificado originalmente, sólo que se repiten cada vez que se corrige un resultado erróneo.

Las pruebas de integración concluyen una vez que se hayan integrado todos los módulos del programa. El resultado de las mismas será documentado en el Expediente de pruebas y los errores obtenidos deben ser recogidos en el registro de defectos de las pruebas.

3.4.5 PRUEBAS DE SISTEMA

3.4.5.1 Prueba de Funcionalidad

- Descripción: Esta prueba se encarga de verificar que el software cumpla con los requerimientos funcionales definidos.
- Objetivo: Evaluar la funcionalidad del sistema funcionando como un todo.
- Técnica: Se van a realizar pruebas fijando su atención en la validación de las funciones, métodos, servicios definidos, así como verificar las habilidades del programa para manejar grandes cantidades de datos, tanto como entrada, salida o residentes en la Base de Datos.
- Responsable de ejecutar la prueba: Probador.
- Responsable de diseñar la prueba: Diseñador de Pruebas.
- Comienzo de la prueba: Estas pruebas comienzan una vez que se hayan probado e integrado todos los módulos que conforman al software.
- Procedimiento para realizar la prueba: El diseñador es el encargado de diseñar los casos de pruebas para medir la funcionalidad del sistema. El administrador de pruebas se encarga de aprobarlos y para que luego el probador los ejecute. Este debe verificar las habilidades del sistema de manejar grandes cantidades de datos, tanto como entrada, salida o residentes en la Base de Datos en caso de que contenga y luego concluir las pruebas documentando todos los resultados que se obtengan.

3.4.5.2 Prueba de Rendimiento

- Descripción: Esta prueba se encarga de verificar que el software cumpla con su rendimiento en tiempo de ejecución.
- Objetivo: Evaluar cuales son los límites del sistema, ya sean a nivel de software o hardware. Medir el rendimiento del producto en uso de memoria RAM, CPU y ancho de banda.
- Técnica: Someter al sistema a elevados niveles de carga.
- Responsable de ejecutar la prueba: Probador.
- Responsable de diseñar la prueba: Diseñador de Pruebas.
- Comienzo de la prueba: Estas pruebas comienzan una vez que se hayan probado e integrado todos los módulos que conforman al software y se tengan creadas todas las condiciones de hardware y software en el lugar donde se llevaran a cabo las pruebas.

- Procedimiento para realizar la prueba: El diseñador es el encargado de diseñar los casos de pruebas de rendimiento, obteniendo la mayor información de la especificación de casos de uso. El administrador de pruebas se encarga de aprobarlos y para que luego el probador verifique, mediante la ejecución de los mismos, que el sistema funcione correctamente en tiempo de ejecución. A menudo pueden usarse instrumentos de software y/o hardware para medir la utilización de recursos. El probador debe ejecutar tareas durante varias horas seguidas para verificar si se afecta la memoria o el rendimiento del sistema al estar bajo las mismas condiciones por tiempo indefinido. Además debe someter al sistema a elevados niveles de carga y medir el uso que hace el mismo del CPU (Central Processing Unit) y de la RAM (Random-Access Memory). Luego de esto el probador documenta los problemas detectados en la tabla de No Conformidades que contiene la plantilla de Diseño de casos de Pruebas propuesta por la universidad.(Ver Anexo 4)

3.4.5.3 Prueba de Resistencia

- Descripción: Prueba que se centra en encontrar los límites del sistema y asegurar que tras un fallo este se recupera sin causar graves problemas.
- Objetivo: Verificar cómo soporta el sistema más interrupciones de las normales. Comprobar cómo reacciona el sistema ante excesivas búsquedas de datos residentes en disco (Cargar una partida, ver el score).
- Técnica: Forzar cada componente de forma aislada más de lo que la aplicación podría experimentar en condiciones normales y observar si hay problemas evidentes.
- Responsable de ejecutar la prueba: Probador.
- Responsable de diseñar la prueba: Diseñador de Pruebas.
- Comienzo de la prueba: Estas pruebas comienzan durante la Fase de Construcción del producto, una vez que se hayan probado e integrado todos los módulos que conforman al software y se tengan creadas todas las condiciones de hardware y software en el lugar donde se llevaran a cabo las pruebas.
- Procedimiento para realizar la prueba: El Diseñador de Pruebas debe diseñar los casos de pruebas a partir del análisis de los riesgos del software realizado. Estos casos de pruebas deben recoger si el sistema en algún momento deja de operar o si es capaz de:

  - Procesar cierta cantidad de datos.
  - Trabajar con un disco que no este a su completa capacidad.
  - Soportar varios accesos simultáneos.

El Administrador de pruebas debe aprobar estos casos de pruebas y luego el probador es el encargado de ejecutarlos. Este debe someter la aplicación a altos niveles de carga y analizar el comportamiento del sistema ante tales condiciones, luego debe documentar los resultados de estas pruebas.

3.4.5.4 PRUEBAS DE INTERFAZ DE USUARIO

- Descripción: Son caracterizadas por otorgarle al usuario un papel principal, ya que estos son los principales a la hora de determinar el grado de usabilidad del producto.
- Objetivo: Verificar que los usuarios finales están satisfechos con la interfaz y que puedan usar el software de forma fácil.
- Técnica: Medir la usabilidad del software mediante la observación y la aplicación de encuestas.
- Responsable de ejecutar la prueba: Usuarios finales.
- Responsable de diseñar la prueba: Diseñador de interfaz de usuario, Diseñador de pruebas.
- Comienzo de la prueba: Durante la Fase de Transición del producto, es decir, una vez terminado el software y liberada la versión Beta del mismo.
- Procedimiento para realizar la prueba: Se le proporciona al usuario prototipos del producto final y se le observa, se le pide que comente sus pensamientos en voz alta mientras juega, se le debe observar mientras trabaja con el prototipo del producto y luego se le hacen preguntas sobre sus opiniones al utilizar el juego y sobre las posibles insatisfacciones que haya podido encontrar. Las preguntas que se le van a hacer al usuario se pueden redactar en forma de encuesta. Generalmente los usuarios finales de los juegos son niños, por lo cual hay que elaborar las preguntas de forma clara y sencilla. Se debe tener en cuenta que esta prueba esta enfocada a comprobar la usabilidad del software por lo que es aconsejable que se incluyan preguntas como:

¿Para comenzar a jugar necesitaste que alguien te explicara como hacerlo o aprendiste de forma fácil?

¿Hubo alguna parte del juego en la que no pudiste seguir avanzando porque no sabías como?

¿Te gusta jugar, encuentras el juego atractivo?

¿Hay algo en específico que no te guste del juego?

3.4.6 PRUEBAS DE ACEPTACIÓN

- Descripción: La prueba está enfocada a comprobar que el producto cumple las expectativas del cliente
- Objetivo: El objetivo de estas pruebas es que el cliente valide el software desarrollado.
- Técnica: Se verifica si el software cumple con los requisititos definidos por el usuario, utilizando las técnicas de caja negra. Se comprueba la correspondencia entre lo que se encuentra en la aplicación y lo descrito en el documento, así como el manual de instalación y el manual de usuario.
- Responsable de ejecutar la prueba: Clientes o usuarios finales.
- Responsable de diseñar la prueba: Cliente, diseñador de casos de prueba.
- Comienzo de la prueba: Durante la Fase de Transición del producto, cuando el software este terminado y ya se le hayan aplicado las pruebas del sistema.
- Procedimiento para realizar la prueba: El cliente es el encargado mayormente de diseñar y ejecutar las pruebas de aceptación. Se debe verificar que se obtengan resultados esperados si se introducen datos válidos, por lo que se debe ejecutar cada caso de uso y flujo del caso de uso con datos válidos e inválidos. Durante estas pruebas se verifica además que cada requisito del negocio se haya implementado correctamente.

3.5 RECURSOS REQUERIDOS

3.5.1 RECURSOS HUMANOS

Los roles y las responsabilidades definidas para este equipo de pruebas, están pensadas para realizar un desarrollo del proceso de pruebas más eficiente y eficaz. Para llevar a cabo las pruebas es necesario que este equipo este compuesto por al menos un profesor que desempeñe el rol de Administrador de Pruebas, dos estudiantes que desempeñen los roles de diseñadores de pruebas, uno que juegue el rol de ingeniero de componentes y al menos dos probadores.

Rol	**Responsabilidades**	**Mínimos recomendados**
Administrador de pruebas	• Asegurar la calidad en el proceso de desarrollo de software. • Asegurar que la aplicación	1 Profesor

	producida se ajusta a las especificaciones y está razonablemente libre de errores. • Proporcionar una metodología para realizar las pruebas. • Coordinar las pruebas de calidad internas, las pruebas de aceptación del cliente y pilotos de conjunto con el Líder de Software y Calisoft. • Evaluar los resultados que se obtienen en las pruebas de calidad	
Diseñador de Pruebas	• Diseñar los casos de prueba. • Evaluar y documentar el resultado de las pruebas realizadas al software. • Definir listas de chequeo.	2 estudiantes
Ingeniero de Componentes	• Elaborar los componentes de prueba para los procedimientos que puedan ser automatizados.	1 estudiante
Probador	• Ejecutar las pruebas diseñadas. • Anotar los resultados obtenidos.	2 estudiantes

3.5.2 RECURSOS SOFTWARE

En esta sección se deben describir los recursos de software necesarios para desarrollar las pruebas de manera exitosa. Estos recursos varían en dependencia de las características del juego.

Recurso	Descripción
Sistema Operativo	Sistema operativo en el que se vaya a trabajar.
Entorno integrado de desarrollo	Herramientas en las que desarrolla.
Herramientas de Automatización	Herramientas para automatizar las pruebas.

Entorno integrado de desarrollo:

- Microsoft Visual Studio .Net 2003 / IDE desarrollado por Microsoft.
- Rational Rose 2003 Enterprise Edition / Herramienta de modelado.
- Radiant / Compilador mapas 3D.
- C++ / Lenguaje de programación Orientado a Objetos.
- OpenAL / Librería para el trabajo con sonido 3D.
- Microsoft Office / Herramienta para editar Documentos.
- Visual Paradigm / Herramienta de Modelado.

Herramientas de Automatización:

C++Test : Pruebas de unidad automatizadas, producto de análisis de código estándar que mejora la fiabilidad de código C/C++, su funcionalidad, su seguridad, su portabilidad y mantenibilidad. Genera automáticamente y ejecuta pruebas unitarias para una verificación instantánea, permitiendo a los usuarios retocar y ampliar estas pruebas según sus necesidades.

IBM- Rational PurifyPlus: Conjunto de herramientas de análisis automatizado en tiempo de ejecución para mejorar la fiabilidad y el rendimiento de las aplicaciones basadas en Windows. Incluye detección de corrupción de memoria, detección de pérdidas de memoria, parametrización del rendimiento de aplicaciones y análisis de cobertura del código. No requiere acceso al código fuente y, por lo tanto, puede utilizarse con bibliotecas de terceros, además del código propio.

3.5.3 RECURSOS HARDWARE

En esta sección se deben describir los recursos necesarios para el desarrollo de las pruebas. A continuación se exponen los recursos mínimos a tener en cuenta a la hora de realizar las pruebas. Como se esta tratando con juegos de realidad virtual, en algunos casos será necesario el uso de periféricos de consolas y PC, los cuales permiten el manejo o control del juego, tales como joystick, cascos, entre otros.

Recurso	Cantidad	Actividad
PC	1	Planificación de las Pruebas
PC	1	Diseño de las Pruebas
PC	2	Ejecución de las Pruebas

3.6 VALIDACIÓN DE LA PROPUESTA

Con el objetivo de llevar a cabo el proceso de validación del trabajo de diploma Estrategia de Pruebas para juegos de Realidad Virtual fueron seleccionados siete expertos. Los mismos fueron seleccionados por estar estrechamente relacionados con el proceso de desarrollo de juegos de Realidad Virtual y con la calidad de software.

Ing. Julián Valdés Santiuste:

Ingeniero en Ciencias Informáticas. Graduado en Julio de 2006. Profesor del Departamento de la especialidad desde septiembre de 2006; Profesor Adiestrado. Profesor de Práctica Profesional 1. Durante el curso 2007-2008 se ha desempeñado como presidente del tribunal de tesis de calidad de software.

Ing. José Manuel Pardo Matos:

Ingeniero en Ciencias Informáticas. Graduado en Julio de 2007. Profesor del Departamento de Ciencias Básicas desde septiembre de 2007; Profesor Adiestrado. Asesor de Calidad de la Facultad 5. Actualmente se encuentra cursando la maestría en Calidad de Software. Ha cursado varios postgrados durante su etapa de adiestramiento entre los que están Docencia e Innovación Universitaria, Auditoría TIC, Técnicas Avanzadas de desarrollo de SW, Métricas de SW, Validación y Verificación, Monitoreo y Control de Proyectos. Participó en UCIENCIA 2007.

Ing. Yulier Casas Estrada:

Ingeniero en Ciencias Informáticas, profesor adiestrado que imparte asignaturas de Programación, con dos años de graduado. Por más de un año se ha desempeñado como líder del proyecto Juegos Cneuro, el cual pertenece al polo de Realidad virtual. Ha participado como ponente en eventos nacionales y en dos internacionales, en todos los casos sus trabajos han sido publicados

Ing. Yoander Cabrera Díaz:

Ingeniero en Ciencias Informáticas, profesor adiestrado que imparte asignaturas de Ingeniería de Software. Graduado en Julio de 2007. Actual líder del proyecto Juegos de Consola, el cual pertenece al polo de Realidad virtual. Ha presentado publicaciones en UCIENCIA en las que ha obtenido la distinción de destacado. Actualmente se encuentra cursando un diplomado de Realidad Virtual

Ing. Roig Calzadilla Diaz.

Especialista de la Dirección de Calidad de Software. Infraestructura Productiva (IP). Graduado de Ingeniería en Ciencias Informáticas, en el curso 2006-2007 en la Universidad de las Ciencias Informáticas, actualmente se encuentra laborando en el mismo centro como instructor recién graduado, y realizando además el cargo de Especialista de Calidad de la Dirección de la Infraestructura Productiva (IP). Posee grandes experiencias en eventos y trabajos que se relacionan con el tema de la Calidad de Software. Su tesis de grado estuvo relacionada con la definición de los procesos de pruebas en el Laboratorio de Calidad. Participó como ponente en el evento internacional JIISIC 2008 celebrado en Ecuador, con un trabajo relacionado con pruebas de software. Realizó en Venezuela pruebas como asesor de Calisoft. Además participó en la 7ma semana de tecnología (FORDES) y presenta dos publicaciones, una nacional y otra internacional.

Ing. Dagoberto Marrero López

Ingeniero en Ciencias Informáticas, profesor adiestrado que imparte asignaturas de Máquinas Computadoras. Graduado en Julio de 2007. Actual líder del proyecto Compilación de Juegos, el cual pertenece al polo de Realidad virtual.

Ing. Yanoski Rogelio Camacho Román:

Ingeniero en Informática. Graduado de la CUJAE en el 2004 (Instituto Superior Politécnico José Antonio Echeverría). Cuatro años de experiencia en los gráficos y la Realidad Virtual, Líder de un proyecto de Realidad Virtual desde el 2004 (Herramientas de desarrollo para sistemas de realidad virtual). Diplomado en Diseño y Programación de Videojuegos de la Universidad Abierta de Catalunya. Premio del rector 2008 en el proyecto.

Los mismos valoraron la estrategia usando como guía los siguientes criterios:

1. ¿Considera Ud. que el Flujo de trabajo propuesto para el desarrollo de las pruebas es correcto?
2. ¿Considera Ud. que las pruebas propuestas son suficientes?
3. ¿Considera Ud. Que los procedimientos para realizar las pruebas son correctos?
4. ¿Considera Ud. que los roles propuestos para el diseño y ejecución de las pruebas son los adecuados?

Ing. Julián Valdés Santiuste:

Yo pienso que el Flujo de trabajo propuesto para el desarrollo de las pruebas es correcto porque cumple con el flujo propuesto por RUP desde el punto de vista muy particular de las tesistas donde existe de algún modo una planificación de la prueba, un diseño de la prueba, una implementación de la prueba, una realización de prueba de integración, una realización de prueba de sistema y finalmente una evaluación de la prueba. Creo que las pruebas propuestas son suficientes debido a que se utilizan los tipos de pruebas mas adecuados en los lugares claves dentro del proceso de desarrollo del software y que de manera general los procedimientos establecidos siguen lo que propone RUP para realizar una prueba. Los roles propuestos para el diseño y ejecución de las pruebas son los adecuados, están definidos correctamente teniendo en cuenta el tipo de prueba que se va a realizar y quién es el más apropiado para la realización de dicha prueba.

Ing. José Manuel Pardo Matos:

Creo que el flujo de trabajo propuesto para desarrollar las pruebas es correcto, ya que define de manera detallada las tareas a realizar durante la planificación, ejecución y control de las pruebas. Los resultados de las pruebas, cualesquiera que estas sean nunca aseguran que un producto software está libre de errores. No obstante, realizar pruebas asegura la calidad del producto a través de la eliminación de fallas, por lo que considero que son suficientes en función de las características de la propuesta. En los procedimientos propuestos para cada tipo de pruebas se definen de manera general el ambiente necesario para realizar las pruebas, las entradas y las salidas para cada caso. En mi opinión, son correctos los procedimientos y roles propuestos.

Ing. Yulier Casas Estrada:

Opino que el flujo de trabajo que se propone para llevar a cabo las pruebas es correcto, puesto que describe detalladamente las actividades a realizar, es importante que este proceso se lleve a cabo dentro del proyecto en estrecha relación con los desarrolladores, las pruebas que se proponen se ajustan bien al proceso aunque en este sentido se puede seguir profundizando en las pruebas del diseño gráfico. En mi criterio, los procedimientos y roles propuestos son correctos.

Ing. Yoander Cabrera Díaz:

Creo que Todas las actividades que se definen en el flujo de trabajo son correctas, de forma general las pruebas que se proponen son adecuadas así como sus procedimientos y roles, pienso que la estrategia de pruebas que se propone se ajusta a los juegos de Realidad Virtual aunque podían tenerse en cuenta además las pruebas de diseño gráfico al llevar a cabo este proceso. Considero que es de vital importancia la definición de los roles de calidad dentro del proyecto y que además es necesario que se capacite y supervise el trabajo de los mismos. Considero que es muy importante que se aplique dicha estrategia puesto que en la facultad todavía tenemos que avanzar mucho con respecto al tema de la calidad.

Ing. Roig Calzadilla Díaz:

Considero que el Flujo de trabajo propuesto para el desarrollo de las pruebas es correcto. Las pruebas propuestas son suficientes y ayudan bastante a realizar verificaciones durante el proceso de desarrollo. Los procedimientos establecidos satisfacen las necesidades de las mismas, sus roles son adecuados y resuelven el problema actual en las pruebas.

Ing. Dagoberto Marrero López:

Yo pienso que el Flujo de trabajo propuesto para el desarrollo de las pruebas es correcto porque cumple con las actividades que se establecen para el desarrollo de las pruebas. Se encuentran incluidos la planificación, el diseño, la implementación de las pruebas y por último la evaluación. Considero que las pruebas propuestas son suficientes y que de manera general los procedimientos establecidos siguen lo que propone RUP para realizar una prueba. Los roles que se proponen para el diseño y ejecución de las pruebas son los adecuados y son definidos acertadamente teniendo en cuenta el tipo de prueba que se va a realizar.

Ing. Yanoski Rogelio Camacho Román:

En mi criterio el flujo de trabajo que se propone es correcto, las pruebas se ajustan a este tipo de software y opino que son suficientes, sus roles y procedimientos son correctos y están definidos de forma adecuada a partir de lo que propone RUP

3.6.1 CONCLUSIONES GENERALES DE LA VALIDACIÓN.

De manera general los expertos en Realidad Virtual y calidad de software que emitieron su criterio a cerca de la investigación realizada, llegaron al consenso de que la propuesta se ajusta perfectamente a las necesidades que presentan actualmente los juegos de Realidad Virtual. El flujo de trabajo que en ella se presenta y los procedimientos para realizar las pruebas, están desarrollado correctamente teniendo en cuenta lo descrito en El Proceso Unificado de desarrollo de Software (RUP). Considera además que las pruebas propuestas son las necesarias para aplicarlas en este tipo de proyecto y en uno de los casos se emitía la posibilidad de agregar en ciertos casos las pruebas de diseño gráfico, para mejorar el proceso de pruebas a este tipo de software. Globalmente los expertos calificaron la propuesta de muy buena, y argumentaron que de aplicarse, mejoraría notablemente el proceso de desarrollo y de pruebas del software de juegos de Realidad Virtual.

CONCLUSIONES

Los resultados del presente trabajo dan cumplimento al principal objetivo del mismo que ha sido establecer una estrategia de pruebas para Juegos de Realidad Virtual. Los principales resultados obtenidos en este trabajo se relacionan a continuación:

Se realizó un estudio de la situación actual de los proyectos de juegos de Realidad Virtual de la facultad 5 evidenciando que no está establecida una estrategia de pruebas que verifique el nivel de calidad del proceso y los productos resultantes.

Se trazó una estrategia de pruebas para aplicar en este tipo proyectos, que consta de las siguientes actividades: revisión al guión técnico, revisión a los requerimientos, pruebas de unidad, pruebas de integración, pruebas de sistema y pruebas de aceptación, así como la propuesta de un flujo de trabajo para realizar estas actividades.

Se definieron los tipos de pruebas a aplicar a los Juegos de realidad virtual para cada nivel de Pruebas. Para las pruebas unitarias se definen las pruebas de funcionalidad y de caja blanca, para las de integración se definen las pruebas de funcionalidad. En el caso de las Pruebas de Sistema se definen como tipos de pruebas a aplicar las pruebas de funcionalidad, resistencia, rendimiento e interfaz de usuario y finalmente las pruebas de aceptación.

RECOMENDACIONES

Se recomienda:

- Poner en práctica la propuesta de la estrategia de pruebas en los proyectos de juegos de Realidad Virtual de la Facultad 5.
- Realizar un estudio de las herramientas automatizadas de prueba para su aplicación en los proyectos productivos de la facultad, ya que es aconsejable su uso puesto que conlleva al ahorro de tiempo, a reducir el esfuerzo requerido, y a aumentar la calidad del producto.
- Capacitar a todo el personal del proyecto implicado en el aseguramiento y control de la calidad del software.
- Evaluar y controlar con sistematicidad los procesos relacionados con las pruebas y sus resultados.

BIBLIOGRAFÍA

1. **8402, ISO.** UNE 66-001-92.

2. **IEEE.** *Standard Glossary of Software Engineering Terminology .* 1990.

3. **Pressman, Roger S.** *Ingeniería de software, un enfoque práctico.* 5ta edición. Madrid : s.n., 2001.

4. **Pressman, Roger S.** *Ingeniería del Software. Un Enfoque Práctico.* Cuarta Edición. s.l. : Editorial Mc Graw-Hill, 1998. p. 581 páginas. ISBN 84-481-1186-9..

5. ***ISO/IEC 12207*** *Procesos del ciclo de vida del software.* 2005.

6. **LOVELLE, J. R. C.** *Aseguramiento de la calidad.* 1999.

7. **MENDOZA, L. E.** Sistemas de Información III. *www.prof.usb.ve.* [Online] http://prof.usb.ve/lmendoza/Documentos/PS-6117%20(Teor%EDa)/PS6117%20Calidad%20del%20Software.pdf.

8. **Pressman, Roger. S.** *Ingeniería de software. Un enfoque práctico.* 2005.

9. **J, H. MONSALVE.** Sistemas de Inormacion III. *www.prof.usb.ve.* [Online] 1998. http://prof.usb.ve/lmendoza/Documentos/PS-6117%20(Teor%EDa)/PS6117%20Calidad%20del%20Software.pdf.

10. **Myers, Glen.** *The Art of Software Testing.* 1979. p. 568 pages. ISBN-10: 0471078786.

11. **Ivar jacobson, Grady Booch,James Rumbaugh.** *El Proceso Unificado de Desrrollo de Software.* Madrid : Addison Wesley, 2000. 84-7829-036-2.

12. **Charles P. Schultz, Robert Bryant and Tim Langdell.** *Game Testing All in One.* 2005. p. 520 pag. ISBN:1592003737.

13. **Fernández-Medina, Dr. Eduardo.** Alarcos. [Online] [Cited: Mayo 6, 2007.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf..

14. **César F. Acebal, Juan M. Cueva Lovelle.** *Extreme Programming (XP): Un nuevo método de desarrollo de software.*

15. **www.fimte.fac.org.ar.** [Online] [Cited: Enero 25, 2008.] www.fimte.fac.org.ar/ayuda/glosario.htm.

16. **www.futurovenezuela.org.** [Online] [Cited: Enero 25, 2008.] www.futurovenezuela.org/TH/glosario.htm.

17. **www.portal-uralde.com.** [Online] [Cited: Enero 25, 2008.] www.portal-uralde.com/dicr.htm.

18. **www.jcyl.es.** [Online] [Cited: Enero 25, 2008.] www.jcyl.es/jcyl-client/jcyl/files/Glosario_de_T%C3%A9rminos_de_la_Socidad_de_la_Informaci%C3%B3n.

19. **members.fortunecity.com.** [Online] [Cited: Enero 25, 2008.] members.fortunecity.com/miprofe/archivos/TerminosComputacionales.htm.

20. **TENDENCIAS Y NUEVAS TECNOLOGIAS-EMPRESA.** *Realidad Virtual y second life.* [Online] [Cited: Enero 15, 2008.] http://tecnologia-empresa.blogspot.com/2008/01/realidad-virtual-y-second-life.html.

21. **video-animation.** [Online] [Cited: Febrero 6, 2008.] (http://www.video-animation.com/flash_18.php).

22. **Jacobson, Ivar.** *Aspect-Oriented Software Development With Use Cases.* Pan-Wei Ng : Addison-Wesley, 2004. ISBN 0321268881 .

23. **Addison-Wesley, Kent Beck.** *Extreme Programming Explained.* 2000 . 201-61641-6.

24. **César F. Acebal, Juan M. Cueva Lovelle.** *Extreme Programming (XP): Un nuevo método de desarrollo de software.*

25. **Pruebas de software.** *http://lml.ls.fi.upm.es.* [Online] [Cited: Mayo 5, 2008.] http://lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt#284-1-pruebasdesoftware.

26. **Diseño de Pruebas** . *http://www.willydev.net.* [Online] [Cited: Mayo 5, 2008.] http://www.willydev.net/InsiteCreation/v1.0/descargas/oguzman-diseno_pruebas.pdf.

27. **Juan Carlos Parra Márquez, Rodrigo García Alvarado,Iván Santelices Malfanti.** *Introduccion Practica a la Realidad Virtual.*

28. **http://lsi.ugr.es.** [Online] [Cited: Mayo 16, 2008.] http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/fases/planificacion_pruebas.php.

29. **http://lsi.ugr.es.** [Online] [Cited: Mayo 16, 2008.] http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/fases/disegno_pruebas.php.

30. **Herramientas para el entorno de pruebas.** *http://www.als-es.com.* [Online] [Cited: Marzo 16, 2008.] http://www.als-es.com/home.php?location=herramientas/entorno-pruebas.

31. **Verma, Amit.** Optimización de estrategias de prueba durante el diseño. *amit.verma@teradyne.com.* [Online]

32. ***PROPUESTA DE UNA METODOLOGÍA DE DESARROLLO DE SOFTWARE EDUCATIVO BAJO UN ENFOQUE DE CALIDAD SISTÉMICA.*** **María Gabriela Díaz-Antón, María Angélica Pérez, Anna C. Grimmán, Luis E. Mendoza.** Universidad Simón Bolívar (USB), Caracas, Venezuela : s.n.

33. ***Introducción a Extreme Programming.*** **Escribano, Gerardo Fernández.** 2002.

34. **IEEE-STD-830.** *ESPECIFICACIONES DE LOS REQUISITOS DEL SOFTWARE.* 1998.

35. ***Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionale****s.* **Anna. C Grimán, María Pérez, Luis. E Mendoza.** Laboratorio de Investigación de Sistemas de Información (LISI), Departamento de Procesos y Sistemas, Universidad Simón Bolívar : s.n.

36. *implementación de una metodología y herramientas de pruebas para el grupo de desarrollo de software.* **Deisy Mosquera Vizuete, Jenny Maria Cuenca Aguilar, Cesar Augusto Hidalgo Ojeda.**

37. ***Acerca de las estrategias de Pruebas de Software.*** **Rebaza, Jorge Carlos Valverde.** Universidad Nacional de Trujillo, Escuela Académico Profesional de Informática : s.n.

38. **John Viega, John McManus.** La Importancia de la Prueba del Software. *galeon.com.* [Online] http://www.galeon.com/neoprogramadores/tioswt.htm.

39. **Acuña, César Javier.** *Pruebas de Software.* s.l. : Universidad Rey Juan Carlos.

40. **UCI.** Direccion de Calidad de Software. *http://calidadsoft.prod.uci.cu/.* [Online]

41. **Solutions, BAM.** FUNCTIONAL TESTING . *BAM Solutions.* [Online] 2008. http://www.bamsol.com/bamsol/index.php?option=com_content&task=blogcategory&id=26&Itemid=40.

ANEXOS

Anexo 1: Entrevista

Nombre del proyecto_____.

¿Conoce Ud. las metodologías de Software que existen?

SI_____ NO_____ Cuales_____

¿En el proyecto se trabaja siguiendo alguna metodología?

SI_____ NO_____ Cuales_____

¿Conoce Ud. las pruebas que propone la metodología utilizada en su proyecto?

SI_____ NO_____ Cuales_____

¿Existe alguna estrategia de pruebas en el proyecto?

SI_____ NO_____

¿En su proyecto se le realizan revisiones al software?

SI_____ NO_____

¿En su proyecto se le realizan pruebas al software?

SI_____ NO_____

¿Que tipo de pruebas le realizan al software?

____Pruebas Unitarias

____Pruebas de Integración

____Pruebas del Sistema

____Pruebas de Aceptación

¿En que momento del desarrollo de software comienza Ud. a realizar cada una de las pruebas que se realizan?

¿En que momento del desarrollo de software termina Ud. de realizar cada una de las pruebas que se realizan?

¿Documenta Ud. los datos históricos de cada una de estas pruebas?

_____.

De estas pruebas, cuales Ud. considera que son más importantes.

____Pruebas Unitarias

____Pruebas de Integración

____Pruebas del Sistema

____Pruebas de Aceptación

¿Cuantas personas integran el equipo de calidad de su proyecto?

_____.

¿Qué persona dentro del proyecto es la encargada de planificar las pruebas (Rol)?.

_____.

¿Qué persona dentro del proyecto es la encargada de realizar las pruebas (Rol)?.

_____.

¿Qué persona dentro del proyecto es la encargada de documentar las pruebas (Rol)?.

_____.

¿Además de las pruebas, en su proyecto se realizan otras actividades relacionadas con el aseguramiento y el control de la calidad?

_____.

¿Que características hacen especiales a los juegos de Realidad Virtual según su criterio?

_____.

Anexo 2: Lista de Chequeo del guión

Proyectos de Juegos de Realidad Virtual

Lista de Chequeo para Guiones Técnicos

Versión 1.0

Control de Versiones

Fecha	Versión	Descripción	Autor

ELEMENTO A PROBAR	SI	NO	OBSERVACIONES
Aspectos Generales			
1-¿El guión contiene el nombre del producto?			
2-¿El guión contiene Historial de versiones?			
3-¿El guión esta redactado de forma coherente?			
4-¿Están claramente definidas todas las funcionalidades que realiza el juego?			

Mapa de Navegación			
1-¿El guión contiene Mapa de navegación?			
2-¿El mapa de navegación contiene todas las pantallas que constituyen el juego?			
3-¿El orden de navegación es correcto?			
Ventanas Emergentes			
1-El guión contiene el nombre de todas las ventanas emergente que aparecen en el juego?			
2--El guión contiene el momento en que se deben mostrar cada una de las ventanas.			
3--¿El guión contiene el diseño de la interfaz de las ventanas.			
4--¿El guión contiene la descripción detallada de las acciones a ejecutar cuando emerge la ventana?			

Pantallas			
1-¿Se tienen definidos identificadores para cada pantalla?			
2-¿Están claramente descritas las funcionalidades de las pantallas?			
3-¿Están presentes todos los recursos que posee la pantalla? • Imágenes. • Animaciones. • Sonidos. • Locuciones. • Texto. • Otros.			
4- ¿Están claramente definidas todas las acciones que debe realizar el usuario?			
5- ¿Están claramente definidas todas las respuestas que debe dar el sistema?			

Anexo 3: Lista de Chequeo de Requerimientos

<Nombre del Proyecto>

Lista de Chequeo

Versión 1.0

Control de versiones:

Fecha	Versión	Descripción	Autor

LISTA DE CHEQUEO REQUISITOS.

Aunque la validación de requisitos puede guiarse de manera que se descubran errores, es útil chequear cada requisito con un cuestionario. Las siguientes cuestiones representan un pequeño subconjunto de las preguntas que pueden plantearse:

- ✓ ¿El requisito en cuestión es necesario?
- ✓ ¿Esta el requisito claramente definido? ¿Puede interpretarse mal?
- ✓ ¿Se considera factible este requisito?
- ✓ ¿El requisito planteado es correcto?
- ✓ ¿Es el requisito consistente?
- ✓ ¿Se puede considerar como completo a este requisito?
- ✓ ¿Se pudiera catalogar como consistente?
- ✓ ¿Este requisito es verificable?
- ✓ ¿Es independiente del diseño?
- ✓ ¿Este requisito se considera redundante?
- ✓ ¿Tiene asignado un identificador único?
- ✓ ¿Este requisito en su redacción usa frases especulativas o conceptos condicionales?
- ✓ ¿Aparecen frases no completas?

Ayuda para los probadores.

Aunque la validación de requisitos puede guiarse de manera que se descubran errores, es útil chequear cada requisito con un cuestionario. Las siguientes cuestiones representan un pequeño

subconjunto de las preguntas que pueden plantearse

- ✓ Necesario: Si el sistema puede satisfacer las necesidades reales sin el requisito, entonces no es necesario.
- ✓ Claramente definido: Un requisito está claramente definido si, y sólo si, cada requisito declarado tiene solo una interpretación. Como mínimo se requiere que cada característica de la última versión del producto se describa utilizando un único término.
- ✓ Factible: El requisito puede ser desarrollado dentro del tiempo y presupuesto.
- ✓ Correcto: Los hechos relacionados con los requisitos son precisos, técnicos y legalmente posibles.
- ✓ Consistente: El requisito esta escrito de forma simple, no expresa más de una funcionalidad.
- ✓ Completo: El requisito expresa una idea o enunciado en su totalidad y las condiciones bajo las que se aplica el requisito se encuentran declaradas.
- ✓ Consistente: No entra en conflicto con otros requisitos.
- ✓ Verificable: La implementación del requisito en el sistema puede ser probada.
- ✓ Independiente del diseño: El requisito no debe asumir una solución de implementación específica.
- ✓ No redundante: No existe duplicación del requisito.
- ✓ Asignado identificador único: Cada requisito debe tener un identificador único.
- ✓ Uso de frases especulativas: p.e. usualmente, generalmente, típicamente.
- ✓ Localización de conceptos condicionales: p.e. quizá, probablemente.
- ✓ Uso de un único verbo por requisito**:** lo que permite comprobar la existencia de una única necesidad en cada requisito.
- ✓ Uso de frases no completas**:** 'más adelante', 'en un futuro'.

Bibliografía

Young, Ralph R. *The Requirements Engineering Handbook.* Norwood : ARTECH HOUSE, INC, 2004.

"Automated Analysis of Requirement Specifications". Wilson, Rosenberg y Hyatt. International Conference on Software Engineering (ISCE '97), Boston, MA, May 1997.

IEEE. IEEE Recommended Practice for Software Requirements Specifications. IEEE/ANSI. Standard 830–1993, Institute of Electrical and Electronics Engineers, 1993.

Anexo4: Reporte de No Conformidades.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	*<1>*	*<Descripción de la No Conformidad>*	*<Descripción del Aspecto correspondiente>*	*<Etapa de detección del error>*	*<X>*	*<X>*	*<X>*	*[Se coloca el estado de la NC y la fecha cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se reviso* *RA: Resuelta* *PD:Pendiente* *NP:No Procede*	*[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.]*

Anexo 5: Diseño de Casos de Prueba

<Nombre del Proyecto>

Módulo <Nombre del Módulo>

Versión del proyecto

Diseño de Casos de Prueba

<Nombre del Módulo>

Versión del proyecto

Control de versiones:

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

Secciones a probar en el Caso de Uso:

Nombre de la sección	***Escenarios de la sección***	***Descripción de la funcionalidad***	***Flujo Central***
[SC 1: Nombre de la sección]	*EC 1.1: Nombre del Escenario*	*Descripción de la Funcionalidad.*	*Pasos a desarrollar para probar la Funcionalidad que se indicó.*
	EC 1.2: Nombre del Escenario.	*Descripción de la Funcionalidad.*	*Pasos a desarrollar para probar la Funcionalidad que se indicó*
	EC 1.n: Nombre del Escenario.	*Descripción de la Funcionalidad.*	*Pasos a desarrollar para probar la Funcionalidad que se indicó.*

SECCIÓN # 1 A REVISAR

Id del escenario	**Escenario**	**Variable 1** *(Nombre de la variable)*	**Variable 2** *(Nombre de la variable)*	**Variable N** *(Nombre de la variable)*	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1	*Nombre del escenario*	*V*	*V*	*V*	*Se escribe el resultado que se espera al realizar la*	*Se escribe el resultado que se obtiene al*
		V	*V*	*V*		
		V	*V*	*V*		

Elemento	**No**	**No conformidad**	**Aspecto correspondiente**	**Etapa de detección**	**Significativa**	**No Significativa**	**Recomendación**	**Estado NC**	**Respuesta del Equipo Desarrollo**
<Nombre del Elemento>	*<1>*	*<Descripción de la No Conformidad>*	*<Descripción del Aspecto correspondiente>*	*<Etapa de detección del error>*	*<X>*	*<X>*	*<X>*	*[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo*	*[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo*

								con la fecha en que se revisó.] *RA: Resuelta* *PD:Pendiente* *NP:No Procede*	*encontrado o no y en caso de no proceder la no conformidad explica por qué.]*

Anexo 6: ESPECIFICACIÓN DE CASO DE PRUEBA

<Nombre del Proyecto>

Lista de Chequeo

Versión

Especificación de Caso de Prueba

Control de Versiones:

Fecha	Versión	Descripción	Autor

1. **Descripción**

   [Breve descripción del desarrollo y objetivos del caso de prueba]

2. **Elementos a probar**

   [Identificar y describir brevemente los elementos y las características a ser ejercitadas por este caso de prueba.]

   Para cada elemento, considere la siguiente documentación del elemento de prueba:

   a) La especificación de requisitos;

   b) La especificación del diseño;

   c) Guía al usuario;

d) El guía de operaciones;

e) El guía de instalación.

3. Condiciones de ejecución

[Descripción de las condiciones de ejecución que se deben cumplir antes de iniciar el caso de prueba, por ejemplo, que se haya realizado correctamente el login en el sistema...]

4. Entradas

[Descripción paso a paso de la ejecución del caso de prueba]

5. Salidas

[Descripción del resultado que se esperaba de la prueba de test en la aplicación]

6. Configuración de Ambiente de pruebas

6.1 Hardware

[Especificar que características y configuración de hardware se requieren para ejecutar este caso de prueba]

6.2 Software

[Especificar que sistema operativo y aplicaciones son necesarias para la ejecución del caso de prueba]

6.3 Others

[Especificar otros requisitos como personal entrenado, necesidades únicas para el caso de prueba, etc.]

7. Evaluación de la Prueba

[Estado del caso de prueba, que puede ser por ejemplo: propuesta, pendiente de evaluación, realizada y satisfactoria]

Anexo 7: Resumen de la Jornada de Trabajo

Pruebas de *<Nombre de las Pruebas a realizar >*

Resultados Parciales de la Jornada de Trabajo

Fecha: *<dd/mm/aa >*

Proyecto: *<Nombre del Proyecto >*

Artefactos a Probar: *<Nombre del artefacto a Probar >*

Elementos revisados y resultados parciales obtenidos en la Jornada de Trabajo.

Elemento a Revisar	Resultados obtenidos
1. Comportamiento de la Asistencia	*<Se debe reflejar la cantidad de estudiante que debían participar en la actividad, en caso de haber ausentes se debe plasmar las causas, si son conocidas>*
2. Nombre de la Tarea y la Cantidad.	*<Se debe reflejar la cantidad y el nombre de las tareas a realizar, y las que se completaron realmente>*

No conformidades detectadas.

Cant. NC Significativas	Cant. NC No Significativas	Cant. Recomendaciones	Observaciones

Causas de los Resultados Obtenidos Negativamente

Resultados obtenidos de Cada Elemento a Revisar	**Causas**
<Aquí se pone el apesto negativo que afecto el desarrollo de las pruebas o revisiones> ***Nota:*** *Se ponen en una fila distinta cada resultado.*	*<Aquí se ponen las causas o justificación de por qué dicho resultado negativo >*

Condiciones de Trabajo existente.

Recursos Materiales	*<Aquí se pone el estado en que se encuentran los recursos materiales con los que se está trabajando >*
Condiciones ambientales.	*<Aquí se ponen las condiciones ambientales en las que se esta trabajando, si el aire acondicionado funciona correctamente, el estado del laboratorio, si hay ruido, filtraciones, etc.>*
Organización del trabajo	*<Forma en que fue distribuido el trabajo>*

Anexo 8: Reporte Final de No Conformidades

<Nombre del Proyecto>

Módulo <Nombre del Módulo>

Versión del proyecto

No Conformidades Detectadas

Pruebas de <Nombre de las Pruebas a realizar >

Versión 0.0

Control de Versiones:

Fecha	Versión	Descripción	Autor

1. ASPECTOS GENERALES

[Descripción de Aspectos Generales a tener en cuenta a la hora de analizar el resultado de las pruebas, incidencias en el momento de su desarrollo y otros aspectos relevantes.]

ELEMENTOS PROBADOS.

[Descripción general o lista de los Elementos Probados, y otros aspectos importantes a tener en cuenta a la hora analizar las No Conformidades Detectadas.]

ELEMENTOS NO PROBADOS Y CAUSAS.

[Descripción general o lista de los Elementos no Probados, la causa de que no se hayan podido realizar las Pruebas y cualquier otro elemento importante que aporte la información

necesaria para que sean analizadas estas causas y resueltas para la siguiente iteración.]

TABLA DE NO CONFORMIDADES DETECTADAS

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa
<Nombre del Elemento>	*<1>*	*<Descripción de la No Conformidad>*	*<Descripción del Aspecto correspondiente>*	*<Etapa de la detección del error>*		

RECOMENDACIONES

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección
<Nombre del Elemento>	*<1>*	*<Descripción de la No Conformidad>*	*<Descripción del Aspecto correspondiente>*	*<Etapa de la detección del error>*

GLOSARIO

- **Caso de Prueba:** un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular.
- **Ciclo de vida:** Periodo de tiempo que comienza con la concepción del producto de software y termina cuando el producto esta disponible para su uso. Normalmente, el ciclo de vida del software incluye las fases de concepto, requisitos, diseño, implementación, prueba, instalación, verificación, validación, operación y mantenimiento, y, en ocasiones, retirada. Nota: Esta fases pueden superponerse o realizarse iterativamente.
- **Desarrollador:** Un profesional técnico experto del software que está implicado en la definición, el diseño, la puesta en práctica, la evaluación, y el mantenimiento de un sistema de software.
- **Especificación de requisitos de software**: Documentación de requisitos fundamentales (necesarios, esenciales e indispensables) de funcionalidades, rendimiento, restricciones y atributos del software, y sus interfaces externas. Su acrónimo inglés es SRS.
- **Herramienta de prueba:** Sistemas de software y/o de hardware, u otros instrumentos, que se utilizan para medir y evaluar un artefacto del software
- **Informática**: La palabra "Informática" está compuesta por los vocablos información y automatización, y fue empleada por primera vez en el año 1962 por Philipe Dreyfus. Se refiere al conjunto de técnicas destinadas al tratamiento lógico y automático de la información, con el fin de obtener una mejor toma de decisiones.
- **Interfaz de usuario**: Interfaz que permite la comunicación entre un usuario y un sistema, o los componentes de un sistema.
- **Manual de instalación**: Documento que contiene la información necesaria para instalar un sistema o un componente, establecer los parámetros iniciales y preparar el sistema o componente para su uso.
- **Manual de usuario:** Documento que contiene la información necesaria para obtener de un sistema o de un componente los resultados deseados.
- **NC:** No conformidad.
- **p.e:** Por ejemplo.
- **Pruebas:** una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto.

- **Prueba funcional**: Prueba que ignora la mecánica interna de un sistema o un componente y centra la atención sólo en las salidas generadas como respuesta a determinadas entradas y condiciones de ejecución.
- **Recurso:** Los medios físicos, humanos, y económicos necesitados apoyar un proceso, una política, un procedimiento, una meta, o un programa; por ejemplo, materiales, herramientas de hardware y de software, documentos de los estándares, miembros del personal, y fondos.
- **Sistema:** Conjunto de procesos, hardware, software, instalaciones y personas necesarios para realizar un trabajo o cumplir un objetivo.
- **Software:** Los programas de ordenador, procedimientos, y opcionalmente la documentación y los datos asociados que forman parte de un sistema.