Universidad de las Ciencias Informáticas

Facultad 6

Título: "Pruebas de Liberación a la Plataforma de Cálculo Distribuido T-arenal".

Trabajo de Diploma para optar por el título de Ingeniero Informático.

Autores:

Arles Amado Tamayo Rosales

Yosvany Núñez Figueiras

Tutores:

Ing. Carlos Luis Hernández Hernández

Ing. Yuneimy Téllez Pérez

Ciudad de la Habana, junio de 2009.

"Año del 50 Aniversario del Triunfo de la Revolución"

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 firmamos la presente a los ____ días del mes de junio del año 2009.

Arles Amado Tamayo Rosales

_____
Firma del Autor

Yosvany Núñez Figueiras

_____
Firma del Autor

Ing. Carlos Luis Hernández Hernández

_____
Firma del Tutor

Ing. Yuneimy Téllez Pérez

_____
Firma del Tutor

Datos de Contacto

Autores:

Arles Amado Tamayo Rosales
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: aatamayo@estudiantes.uci.cu

Yosvany Núñez Figueiras
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: ynunez@estudiantes.uci.cu

Tutores:

Ing. Carlos Luis Hernández Hernández
Instructor Recién Graduado.
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: chernandez@uci.cu

Ing. Yuneimy Téllez Pérez
Instructor Recién Graduado.
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: ytellez@uci.cu

AGRADECIMIENTOS

Primeramente a nuestros tutores: Carlos y Yuneimy, nuestros guías en esta investigación y quienes demostraron ver este trabajo más allá del deber. Muchísimas gracias por sus consejos y opiniones.

A nuestros profesores a los largo de estos cinco años, quienes han dejado sus huellas en nuestros conocimientos.

Infinitamente A mis padres: Lilian, Arles y Rafael, mis grandes faros en la vida; a quienes le debo el haber llegado hasta aquí y mi espíritu de seguir adelante.

A Onailet, por su amor y apoyo incondicional, demostrándome el verdadero significado de: "dos personas siempre pueden más que una".

A Arodys, el inigualable amigo que todos anhelamos tener, ese con quien podemos contar ciegamente.

A mis inolvidables compañeros del 6X03, quienes acuñaron para mí esa frase: "la universidad es la mejor etapa de la vida".

A mi familia, esa maravillosa y única por la que me tildo de privilegiado al poder contar con su amor y cariño.

A mi mamá, por ser la razón por la que hoy yo pueda cumplir mi sueño y convertir todo su esfuerzo y sacrificio en realidad, por quererme tanto, por creer en mí y estar al lado mío siempre.

A mi PAPÁ Juan, por ser mi ejemplo, por enseñarme a ser todos los días una mejor persona, por guiarme y darme fuerza para lograr lo que me proponga.

A mis abuelos maternos, Elma y Armando que aunque no estén hoy, los llevo siempre en mi corazón y si soy lo que soy, en parte, es gracias a ellos. Ojalá estuvieran presentes y pudieran abrazar y besar a su ingeniero.

A mi novia Tatiana por su comprensión y cariño, su ayuda y por estar al lado mío todo

A mis hermanos, esas pequeñas partes de mí que inconscientemente me hacen crecer, gracias por su sonrisa y permitirme ser su reflejo.

A mis adorados abuelos: Rosa, Martha, Rolando y Primo, esas grandes personitas que me prohíben olvidar que esperan lo mejor de mí.

Y a aquellas personas que de cierto modo estuvieron muy cerca de mí en esta genial etapa y me permitieron contar con su apoyo y compañía.

A todos, de corazón: muchas gracias.

Arles

este tiempo. A toda su familia por toda la atención que me han brindado.

A mi hermanita del alma, por quererme tanto y brindarme siempre su cariño.

A mis tías Juana, Carmén e Idalmis por toda su ayuda a lo largo de este gran camino, a mis primos y a toda mi familia.

A Eduardo mi hermano, a Eric y su familia, a Geno, Raúl, Lenio, Enrique, en fin a todo el grupo por estar ahí siempre conmigo.

A todas las demás personas que me brindaron su apoyo incondicional de una forma u otra, llegue a todos, mi gratitud y mi más puro agradecimiento.

A todos, gracias.

Yosvany

DEDICATORIA

A mi MAMÁ con el amor más grande del mundo, a quien no tengo como expresarle cuán agradecido me siento por su amor, y a quien le debo lo que soy: ***Sienta suyo este triunfo.***

Arles

A mis padres por todos sus esfuerzos que hoy son mis logros, a toda mi familia por toda su ayuda, a mi novia por su amor, a todos mis amigos por su incondicionalidad, a mi grupo, por estar siempre juntos, a todos los que pusieron su granito de arena para poder llegar hasta aquí.

Yosvany

RESUMEN

Pruebas de Liberación es un término usado en la Universidad de las Ciencias Informáticas, y no es más que, un conjunto de pruebas que se realizan a todo software producido en la misma, para avalarlo como apto para entregarse al cliente. El sistema no es el único artefacto a probar, su documentación también forma parte del campo de acción de las pruebas, principalmente aquellos documentos que son entregados junto al producto para guiar y ayudar al cliente a interactuar con el programa. El siguiente trabajo abarca el proceso de pruebas de Liberación a la Plataforma de Cálculo Distribuido T-arenal, planificando primeramente, una estrategia para guiar todas las pruebas, seguido de la ejecución de cada prueba y finalmente la evaluación de los resultados.

Dentro de las pruebas a realizar se encuentran: **Pruebas a la Documentación** que se ejecutan con el objetivo de entregar al cliente una documentación a la altura del producto, bien detallada y libre de errores. **Pruebas Funcionales**, con las que se verifica que el producto responde correctamente a cada funcionalidad establecida por el cliente en forma de requerimientos funcionales. **Pruebas de Seguridad**, para comprobar el nivel de seguridad que brinda el software a la información que manipula. Por último las **Pruebas de Carga,** que permiten conocer el comportamiento del sistema frente a determinada carga de trabajo. Como culminación del proceso, se presentan los resultados arrojados y se evalúa el software a través de una lista de chequeo, según los siguientes parámetros de calidad: Seguridad, Portabilidad, Usabilidad y Confiabilidad.

PALABRAS CLAVES

Pruebas de Liberación, Pruebas a la Documentación, Pruebas Funcionales, Pruebas de Seguridad, Pruebas de Carga, Seguridad, Portabilidad, Usabilidad, Confiabilidad.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El desarrollo de la industria del software en el mundo ha evolucionado a un ritmo acelerado, conjuntamente ha ido aumentado el volumen y complejidad de los productos software, y paralelamente la exigencia de la calidad de estos; sin embargo no se han obtenido los resultados que realmente se esperaban, convirtiéndose así, en un tema crítico para la sociedad actual.

Esta situación se debe a problemas como son: poco entendimiento acerca de las verdaderas necesidades de los clientes, poca experiencia en la gestión y planificación de proyectos, incorrecta aplicación de procesos de desarrollo de software, entre otros. Desde la década del 70, este tema, ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de sistemas informáticos.

Por lo que existen interrogantes que dan lugar a amplias respuestas, las que están estrechamente relacionadas con el concepto de calidad de software: ¿Cómo obtener un software con calidad? ¿Cómo evaluar la calidad del software?

La calidad del software implica un conjunto de cualidades que caracterizan y determinan la utilidad y existencia de un software. Calidad es sinónimo de eficiencia, flexibilidad, confiabilidad, usabilidad, seguridad e integridad; es medible y es imprescindible tener en cuenta su obtención y control durante todas las etapas del ciclo de vida de un software.

El interés por la calidad crece de forma continua a medida que los clientes se vuelven más selectivos y comienzan a rechazar productos poco fiables, o que realmente no dan respuesta a sus necesidades. Esta percepción en el software es el reflejo inherente de los problemas existentes, en pocas palabras, consiste en tratar de cambiar la idea de un simple proceso no sistemático a un proceso determinado por la planificación y la metodología.

Cuba poco a poco ha ido desenvolviéndose en este sentido, ha tomado el desarrollo de una industria de software como una tarea de gran prioridad para el estado, debido a la alta perspectiva económica que posee en el mundo.

Una gran expresión del software cubano, es sin dudas, la Universidad de las Ciencias Informáticas (en lo adelante UCI) creada en el año 2002 y "cuya misión es producir software y servicios informáticos a partir de la vinculación del estudio – trabajo como modelo de formación". (UCI, 2008).

La producción de la UCI se centra en el desarrollo de proyectos destinados a distintas esferas, entre las que se destacan la salud, educación, software libre, bioinformática[1] y procesamiento de imágenes y señales.

Para obtener productos con calidad se creó un laboratorio de pruebas en cada facultad de dicha universidad, los que tienen como objetivo contribuir al aseguramiento de la calidad de los productos que allí se desarrollen, y son los encargados de realizar las pruebas necesarias al software para garantizar su calidad antes de entregarse al cliente.

La Plataforma de Cálculos Distribuido T-arenal es uno de los productos desarrollados en la línea investigativa de Bioinformática. Es un sistema que emplea el tiempo ocioso de las estaciones de trabajo disponibles en una red local para realizar tareas que requieren de un elevado poder de procesamiento, que normalmente necesitarían días, semanas y hasta meses para llevarse a cabo en un solo computador.

La herramienta T-arenal, está conformada por 4 módulos de desarrollo:

- Módulo Montaje de Aplicaciones
- Módulo Servidor
- Módulo Cliente
- Módulo Interfaz de Usuario

Este producto ha llegado a la etapa final de su ciclo de desarrollo, pero no ha pasado por un proceso de pruebas que certifique que está apto para ser entregado al cliente, por lo que no se puede garantizar que está libre de errores, que funcione correctamente en el entorno para el cual fue desarrollado y que responde a cada una de las necesidades del cliente descritas como requerimientos. Por esto se plantea como **problema científico** de la investigación ¿Cómo verificar la calidad de la Plataforma de Cálculo Distribuido T-arenal?

[1] Ciencia que se encarga del estudio, comprensión y análisis de datos biológicos utilizando herramientas informáticas.

Como **objeto de estudio** se define el proceso de realización de pruebas en la industria del software.

El **campo de acción** se enmarca en el proceso de pruebas de Liberación en el laboratorio de calidad de la Facultad 6.

Para dar solución al problema se define como **objetivo general:** realizar pruebas de Liberación a la Plataforma de Cálculo Distribuido T-arenal.

Del objetivo propuesto se derivan los siguientes **objetivos específicos** de la investigación:

- Realizar pruebas a la documentación de T-arenal.
- Diseñar casos de pruebas.
- Realizar pruebas al sistema.
- Documentar los resultados obtenidos en las pruebas.

Para dar cumplimiento a estos objetivos se proponen las siguientes **tareas:**

- Estudio sobre calidad y pruebas de software.
- Realización del Plan de Prueba.
- Revisión del documento Especificación de Requisitos.
- Revisión del documento Descripción de Casos de Uso.
- Revisión del documento Diccionario de Datos.
- Revisión del documento Manual de Usuario.
- Revisión del documento Manual del Desarrollador.
- Revisión del documento Manual de Instalación.
- Diseño de Casos de Prueba.
- Realización de pruebas Funcionales a T-arenal.
- Realización de pruebas de Seguridad a T-arenal.
- Realización de pruebas de Carga a T-arenal.
- Realización de la Plantilla de No Conformidades.
- Evaluación del producto.

El trabajo está estructurado en tres capítulos, descritos a continuación:

En el **Capítulo 1. Fundamentación Teórica** se describen los aspectos más importantes vinculados a la calidad de software. Se exponen conceptos, métodos y procedimientos de pruebas usados en la actualidad; también se incluyen las pruebas a realizar al producto T-arenal para verificar su calidad, así como los métodos, técnicas y herramientas utilizadas.

En el **Capítulo 2. Diseño y aplicación de las pruebas a T-arenal** se presenta la composición del software T-arenal, junto a sus casos de uso y requerimientos. Se muestra la estrategia trazada para guiar las diferentes pruebas y se describe detalladamente el proceso de ejecución de cada prueba.

En el **Capítulo 3. Análisis de los Resultados** se detalla la plantilla de No Conformidades, exponiendo los defectos encontrados durante el período de pruebas a T-arenal. Finalmente se evalúa el software según las siguientes variables de calidad: Usabilidad, Seguridad, Confiabilidad y Portabilidad.

Capítulo 1
Fundamentación Teórica

1.1 Introducción

Los productos software están incrementando continuamente su presencia en muchos aspectos de la vida cotidiana, dotados cada vez de más funcionalidades y más complejos. Este incremento de complejidad trae consigo la posibilidad de un aumento de fallos y errores durante su utilización, que pueden conducir a consecuencias catastróficas en términos económicos y de tiempo. Por lo tanto, existe una necesidad importante de incluir actividades de aseguramiento de calidad durante el proceso de desarrollo y mantenimiento del software.

En este capítulo se exponen conceptos relacionados con la calidad de software, estrategia, niveles de prueba, tipos o técnicas, métodos y herramientas utilizadas en estas pruebas; seleccionando de ello los más convenientes para realizar las pruebas de Liberación al producto T-arenal.

1.2 Calidad de Software

La calidad de software es un problema actual que afecta tanto a los productores de software como a los clientes. Con el aumento de la informatización a escala mundial la demanda de software ha crecido exponencialmente, pero sucede que muchas veces los clientes reciben el software sin haberse completado la etapa de pruebas correspondiente.

La calidad de software es la “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”. (Pressman, 2001).

El Instituto de Ingenieros Eléctricos y Electrónicos definió la calidad de software como: “grado con el que un sistema, componente o proceso cumple con los requerimientos especificados y las necesidades o expectativas del cliente o usuario”. (IEEE, 1990).

Para conseguir una buena calidad del software es esencial establecer un programa de medidas a tomar con respecto a los suministradores. Es también importante utilizar los modelos y métodos apropiados para controlar el proceso de desarrollo.

1.3 Pruebas de software

La prueba de software es uno de los procesos fundamentales dentro del control de calidad del software. “Consiste en la ejecución de un programa bajo ciertos datos de entrada (casos de prueba), para posteriormente comparar las salidas obtenidas con las deseadas (función para la cual fue diseñado)”. (Cosín, 2007).

Según el IEEE, prueba se define como: “Una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente”. (IEEE, 1990).

Se puede decir que la prueba de software es el proceso de establecer la existencia de errores en un programa o sistema; que nunca demuestra que un programa es correcto, siempre es posible que existan errores aun después de la prueba más completa. Las pruebas de software sólo puede demostrar la presencia de errores, no su ausencia.

1.3.1 Objetivos de las pruebas de software

La prueba es un proceso destructivo; se diseña para hacer que el comportamiento de un programa sea distinto del que intentaba su diseñador. Como es una característica humana natural que un individuo tenga cierta afinidad con los objetos que ha construido, el programador responsable de la aplicación del sistema no es la persona más apropiada para probar un programa. Psicológicamente, el programador no querrá destruir su creación, lo cual hará que, de manera consciente o inconsciente elija pruebas que fallan, por lo que no serán adecuadas para demostrar la presencia de errores en el sistema.

Los principales objetivos que se deben tener en cuenta a la hora de realizar cualquier prueba de calidad a un software son:

- Encontrar defectos en el software
  - Una prueba tiene éxito cuando se obtiene al menos un defecto o error.

  - Una prueba fracasa si hay defecto y no lo descubre.
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, con la menor cantidad de tiempo y esfuerzo.
  - La prueba no puede asegurar la ausencia de defectos, solo pueden mostrar que existen.

1.3.2 Flujo de Información de la prueba

Para dar comienzo al flujo de información de la prueba, se proporcionan dos clases de entradas:

- Una configuración del software que incluye la especificación de requerimientos, la especificación de diseño y el código fuente.
- Una configuración de prueba que incluye un plan y procedimiento de prueba, casos de prueba y resultados esperados.

En realidad, la configuración de prueba es un subconjunto de la configuración del software cuando se considera el proceso de ingeniería del software completo.

Se lleva a cabo la prueba y se evalúa los resultados. Es decir, se comparan los resultados de la prueba con los esperados. Cuando se descubren datos erróneos, esto indica la existencia de un error.

A medida que se van recopilando y evaluando los resultados de la prueba, comienza a vislumbrarse una medida cualitativa de la calidad y fiabilidad del software. Si se encuentran con regularidad errores serios que requieren modificaciones en el diseño, se dudará de la calidad y la fiabilidad del software, siendo necesarias posteriores pruebas. Y si el funcionamiento del software parece ser correcto y los errores que se encuentran son de fácil corrección, se pueden sacar dos conclusiones:

- La calidad y la fiabilidad del software son aceptables.
- Las pruebas son inadecuadas para descubrir errores serios.

Finalmente, si la prueba no descubre errores, quedará la sospecha de que la configuración de la prueba no ha sido la más óptima y que los errores están escondidos en el software. Estos defectos serán descubiertos por el usuario y reparados por el profesional en la etapa de mantenimiento.

1.4 Estrategia de pruebas de software

Para aplicar las pruebas al software se debe seguir un conjunto de estrategias para lograr que estas se hagan en el menor tiempo posible y con la calidad requerida, además de lograr que arrojen los resultados esperados.

"Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software". (Pressman, 2001).

La estrategia proporciona un mapa a seguir para el responsable del desarrollo del software, al grupo de control de calidad y al cliente; un mapa que describe los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar esos pasos, cuánto esfuerzo, tiempo y recursos se van a requerir. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de las pruebas y la agrupación y evaluación de los datos resultantes.

1.5 Niveles de pruebas de software

Las pruebas se aplican con objetivos específicos, en diferentes escenarios o niveles de trabajo. Teniendo en cuenta esto, las pruebas se agrupan por niveles de acuerdo a las diferentes etapas del proceso de desarrollo.

1.5.1 Prueba de Unidad

Estas pruebas se consideran una de las primordiales, ya que los resultados obtenidos de ellas repercutirán directamente en la ejecución de las demás pruebas.

Las pruebas de unidad son comprobaciones sobre las **unidades lógicas**[2] de un programa. Se verifica que una unidad funciona correctamente por sí misma, sin tener en cuenta las relaciones que pueda tener con otras partes del sistema.

[2] Partes en que se divide el programa para entenderlo mejor; dígase clases, paquetes, módulos, subsistemas, funciones o cualquier otro mecanismo que nos ofrezca el lenguaje de programación que estamos utilizando.

El modo de operación de este tipo de prueba se basa directamente en concepto de caja blanca, controlando a base de condiciones límites, el buen funcionamiento interno del módulo; esto se refiere al manejo de flujos de datos internos en el módulo controlando las iteraciones, bucles y comparaciones que este realice para verificar todos los posibles caminos que pueda tomar la información recibida. En esta prueba se deben descubrir errores como son:

- Comparaciones entre tipos de datos distintos.
- Operadores lógicos o precedencia incorrecta.
- Igualdad esperada cuando los errores de precisión la hacen poco probable.
- Variables o comparadores incorrectos.
- Terminación de bucles inapropiada o inexistente.
- Fallo de salida cuando se encuentra una iteración divergente.
- Variables de bucles modificadas de forma inapropiadas.

1.5.2 Prueba de Integración

El proceso de integración del sistema implica construir éste a partir de sus componentes y probar el sistema resultante para encontrar problemas que pueden surgir debido a la integración de los componentes. Las pruebas de integración comprueban que estos componentes realmente funcionan juntos, son llamados correctamente y transfieren los datos correctos en el tiempo preciso a través de sus interfaces.

La integración del sistema implica identificar grupos de componentes que proporcionan alguna funcionalidad al sistema e integrar estos añadiendo código para hacer que funcionen conjuntamente. Algunas veces, primero se desarrolla el esqueleto del sistema en su totalidad y se le añaden los componentes, a esto se le denomina integración descendente. De forma alternativa, pueden integrarse primero los componentes de infraestructura que proporcionan servicios comunes, tales como el acceso a base de datos y redes, y a continuación pueden añadirse los componentes funcionales, ésta es la integración ascendente.

La principal dificultad que surge en las pruebas de integración es localizar los errores que se descubren durante el proceso. Existen interacciones complejas entre los componentes del sistema y cuando se descubre una salida anómala, es difícil encontrar la fuente del error. Para hacer más fácil la localización de errores, se utiliza un enfoque incremental para la integración y pruebas del sistema. De

forma inicial, se debe integrar una configuración mínima del sistema y probarlo; luego se agregan componentes a esta configuración mínima y se prueban después de agregados en cada incremento.

Prueba Descendente

Las pruebas descendentes son una parte integral del proceso de desarrollo descendente, en el cual este último inicia con los componentes de los niveles altos y desciende en la jerarquía. Estos tienen la misma interfaz que los componentes pero funcionalidad muy limitada. Después que se programa y prueba el primer componente de nivel alto, se implementan y prueban sus subcomponentes de la misma forma.

Este proceso continúa hasta que los componentes de nivel bajo se implementen y así queda completamente probado el sistema.

Prueba Ascendente

Las pruebas ascendentes comprenden integrar y probar los módulos en los niveles bajos, después asciende por la jerarquía de los módulos hasta que el módulo final se prueba.

Este enfoque no requiere que el diseño arquitectónico del sistema esté completo, por lo que se puede comenzar en una etapa inicial del proceso de desarrollo. Se emplea donde el sistema reutiliza y modifica componentes de otros sistemas.

Las pruebas de integración descendente y ascendente se comparan teniendo en cuenta lo siguiente:

- **Validación arquitectónica**: las pruebas descendentes son susceptibles a descubrir errores en la arquitectura del sistema y en el diseño de alto nivel en las etapas iniciales del proceso de desarrollo.
- **Demostración del sistema**: en el desarrollo descendente se dispone de un sistema funcional limitado en las primeras etapas de desarrollo.
- **Observación de las pruebas**: tanto las pruebas ascendentes como descendentes pueden tener problemas con la observación de la prueba. En muchos sistemas, los niveles más altos del sistema que se implementan primero en un proceso descendente no generan salidas; sin embargo, para probar estos niveles, se les debe forzar a hacerlo. El probador debe crear un entorno artificial para generar los resultados de la prueba. Para las pruebas ascendentes, también

es necesario crear un entorno artificial con el fin de que se pueda observar la ejecución de los componentes de los niveles inferiores.

La selección de una estrategia de integración depende de las características del software y, a veces, de la planificación del proyecto. En general, el mejor compromiso puede ser un enfoque combinado (a veces denominado "prueba de sandwich") que use la descendente para los niveles superiores de la estructura del programa, junto con la ascendente para los niveles subordinados.

1.5.3 Prueba de Validación

Tras la culminación de las pruebas de Integración, el software está completamente ensamblado como un paquete, se han encontrado y corregido los errores de interfaz y puede comenzar una serie final de pruebas del software: la prueba de validación.

La validación puede definirse de muchas formas, pero una simple definición es que la validación se consigue cuando el software funciona de acuerdo con las expectativas razonables del cliente.

Las expectativas razonables están definidas en la especificación de requisitos del software. La especificación contiene una sección denominada «Criterios de validación». La información contenida en esa sección forma la base del enfoque a la prueba de validación.

Criterios de la prueba de Validación

La validación del software se consigue mediante una serie de pruebas que demuestran la conformidad con los requisitos. Un plan de prueba traza las clases de pruebas que se han de llevar a cabo, y un procedimiento de prueba define los casos de prueba específicos en un intento por descubrir errores de acuerdo con los requisitos.

Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen todos los requisitos funcionales, que se alcancen todos los requisitos de rendimiento, que la documentación sea correcta e inteligible y que se alcancen requisitos como son portabilidad, compatibilidad, recuperación de errores, facilidad de mantenimiento, entre otros.

Una vez que se procede con cada caso de prueba de validación, puede darse una de las dos condiciones siguientes:

1) Las características de funcionamiento o de rendimiento están de acuerdo con las especificaciones y son aceptables.
2) Se descubre una desviación de las especificaciones y se crea una lista de deficiencias.

Las desviaciones o errores descubiertos en esta fase del proyecto raramente se pueden corregir antes de la terminación planificada. A menudo es necesario negociar con el cliente un método para resolver las deficiencias.

Revisión de la configuración

Un elemento importante del proceso de validación es la revisión de la configuración. La intención de la revisión es asegurarse de que todos los elementos de la configuración del software se han desarrollado apropiadamente, se han catalogado y están suficientemente detallados para soportar la fase de mantenimiento durante el ciclo de vida del software.

1.5.4 Prueba de Sistema

Las pruebas de sistema coincidirían con las fases finales de las pruebas de integración. Se trata de probar el sistema en su totalidad. El método tradicional es pasar una serie de baterías de pruebas de manera manual por parte del participante con rol de probador. Se describen en una serie de plantillas de manera textual las entradas o pasos a realizar, frente a las salidas esperadas.

“La prueba de sistema, realmente, está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar 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”. (Pressman, 2001).

En estas pruebas se ejercitan condiciones que anteriormente habría sido difícil comprobar, como las interfaces entre subsistemas, la comunicación con el entorno o el cumplimiento de los requisitos no funcionales.

En la Tabla 1.1 se resumen los tipos de pruebas más importantes que se ejecutan en el nivel de sistema y sus objetivos.

Tipos de pruebas	Breve descripción de sus objetivos
Funcionales	Orientadas a detectar si el software implementa adecuadamente las funcionalidades descritas en los requisitos.
De comunicación	Se prueban las interfaces de comunicación entre diferentes componentes del sistema.
Rendimiento	Se determina si los tiempos de respuesta del sistema, tanto en condiciones normales como en condiciones especiales, se encuentran dentro de los límites predefinidos.
Volumen	Dedicadas a probar el software trabajando con grandes cantidades de datos, similares a las esperadas en producción.
Sobrecarga	Se prueba el sistema con cargas masivas de trabajo. Permiten identificar los límites soportables por el software y eliminar comportamientos indeseados en situaciones de sobrecarga.
Disponibilidad	Consisten en probar la disponibilidad del sistema ante fallos de diferentes componentes de la arquitectura, físicos o lógicos.
Usabilidad	Se intenta determinar si el sistema será fácil de utilizar para sus usuarios.
De entorno	Orientadas a probar las interacciones entre el sistema y otros sistemas en su entorno.
Seguridad	Se trata de violar toda protección del sistema, por ejemplo, probando aspectos relativos a la confidencialidad o integridad de la información.

Tabla 1.1 Tipos de pruebas de sistema

1.5.5 Prueba de Implantación

Las pruebas de Implantación se realizan con el software ya instalado en su entorno de operación real y, por tanto, se centran en dos aspectos: los errores derivados de la integración de hardware y software del propio sistema, y la interacción del sistema con el resto de los sistemas de instalación.

Se analizan, primordialmente, requisitos no funcionales, pero también funcionales que dependan fuertemente del entorno operacional real o que requieran la interacción con sistemas que no encontrarán disponibles hasta la fase de implantación.

En concreto, se realizan pruebas de seguridad para verificar si existen errores en los mecanismos de protección dependientes del entorno de operación. También deben realizarse pruebas de rendimiento reales, puesto que en las pruebas de sistema sólo habrán podido comprobarse rendimientos teóricos sobre el entorno de desarrollo. Por último, se examinan los mecanismos de recuperación y copia de seguridad, de modo que se detecten errores en caso de caídas en los servicios software o hardware.

1.5.6 Prueba de Aceptación

Estas pruebas se realizan para que el cliente certifique que el sistema es válido para él. La planificación detallada de estas pruebas debe haberse realizado en etapas tempranas del desarrollo del proyecto, con el objetivo de utilizar los resultados como indicador de su validez: si se ejecutan las pruebas documentadas a satisfacción del cliente, el producto se considera correcto y, por tanto, adecuado para su puesta en producción.

Las pruebas de aceptación son básicamente pruebas funcionales sobre el sistema completo, buscando una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, sino una vez pasada todas las pruebas de integración por parte del desarrollador.

Pruebas Alfa y Beta

Es virtualmente imposible que un encargado del desarrollo pueda predecir cómo un cliente usará realmente el programa, se puede interpretar mal las instrucciones de uso, se pueden usar regularmente extrañas combinaciones de datos y una salida que puede estar clara para el que realiza la prueba pero puede resultar complicada de entender para un usuario normal.

Al momento de construir un software se llevan a cabo una serie de pruebas de aceptación para que el cliente valide todos los requerimientos. Estas pruebas pueden tener lugar a lo largo de semanas o meses, descubriendo así, errores acumulados que podrían degradar el sistema.

Si el software se desarrolla como un producto para muchos clientes, no es práctico realizar pruebas de aceptación para cada uno de ellos. La mayoría de los desarrolladores de productos software llevan a

cabo un proceso denominado pruebas alfa y beta para descubrir errores que parezcan que solo el usuario final pueda descubrir.

La prueba **Alfa** es conducida por un cliente en el lugar de desarrollo. Se usa el software de forma natural, con el encargado del desarrollo mirando por encima del hombro del usuario y registrando errores y problemas de uso. Las pruebas alfa se desarrollan en un entorno controlado.

La prueba **Beta** se lleva a cabo en uno o más lugares de clientes por los usuarios finales del software. A diferencia de la prueba alfa, el encargado del desarrollo normalmente no está presente. Así, la prueba beta es una aplicación en vivo del software en un entorno que no puede ser controlado por el equipo de desarrollo.

El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al equipo de desarrollo. Como resultado de los problemas anotados durante la prueba beta, el equipo de desarrollo del software lleva a cabo modificaciones y así prepara una versión del producto para toda la base de clientes.

1.5.7 Prueba de Regresión

Todos los sistemas sufren una evolución a lo largo de su vida activa. En cada nueva versión se supone que se corrigen defectos o se añaden nuevas funciones o ambas cosas. En cualquier caso, una nueva versión exige una nueva pasada por las pruebas. Si éstas se han sistematizado en una fase anterior, ahora pueden volver a pasarse automáticamente, simplemente para comprobar que las modificaciones no provocan errores donde antes no los había.

El mínimo necesario para usar unas pruebas en una futura revisión del programa es una documentación muy clara.

Las pruebas de regresión son ideales cuando se trata de probar la interacción con un agente externo. Existen empresas que viven de comercializar productos que graban la ejecución de una prueba con operadores humanos para luego repetirla cuantas veces haga falta reproduciendo la grabación. En ambos casos deben monitorizarse las respuestas del sistema, compararlas y avisar de cualquier discrepancia significativa.

≡ **Selección de niveles y técnicas**

Luego del estudio minucioso sobre cada nivel de prueba, los objetivos que persiguen estas y las posibles técnicas a utilizar, se determinó que el nivel de Sistema es el más adecuado para el proceso de pruebas que se realizará.

Esto se decidió principalmente por las características del producto a liberar, el cual cuenta con un módulo que brindará la interfaz visual a través de la cual interactuará el usuario una vez integrado el sistema completamente; las pruebas serán dirigidas a verificar que se han integrado adecuadamente todos los elementos del sistema y que el mismo realiza todas las funcionalidades requeridas.

Dentro de los tipos de pruebas que se pueden aplicar en este nivel, se seleccionaron las siguientes para el proceso de pruebas al producto T-arenal:

- **Pruebas Funcionales:** para verificar que el sistema contiene implementada adecuadamente cada una de las funcionalidades acordadas con el cliente para el software en forma de requisitos.
- **Pruebas de Seguridad:** con el objetivo de verificar el nivel de seguridad que brinda el sistema para proteger la información que almacena; que solamente accedan al sistema los usuarios registrados previamente y que éstos tengan acceso únicamente a la información determinada según su rol.
- **Pruebas de Carga:** estas pruebas se realizarán con el objetivo de conocer los límites de carga de trabajo que soporta el sistema trabajando correctamente, sin llegar a fallar.

1.6 Métodos de prueba de software

♦ **Caja Blanca**

La prueba de Caja Blanca es un método para diseñar casos de prueba usando la estructura de control del diseño procedimental para derivarlos. Mediante los métodos de esta prueba el ingeniero de software puede derivar casos de prueba que garanticen que:

- Se ejercitan al menos una vez todos los caminos independientes de cada módulo.
- Se ejercitan todas las decisiones lógicas en sus caras verdaderas y falsas.
- Se ejecutan todos los bucles en sus límites y con sus límites operacionales.
- Se ejercitan las estructuras de datos internas para asegurar su validez.

Existen varias estrategias que permiten obtener casos de pruebas a partir del código fuente de un programa. A continuación se muestran algunas de las más representativas:

- **Cobertura de caminos**

  Esta estrategia consiste en diseñar casos de prueba suficientes para que se ejecuten todos los caminos de un programa, al menos un caso de prueba por camino. Entiéndase por camino una secuencia de sentencias encadenadas desde la entrada del programa hasta su salida. Para aplicar esta estrategia, debe haberse calculado la cantidad de caminos que posee la estructura lógica que se desea probar.

- **Prueba de bucles**

  Se basan en diseñar casos de prueba en los que los bucles se ejecuten de diferentes maneras. Por ejemplo, pueden plantearse pruebas en las que el cuerpo de un bucle sólo se ejecuta una vez o bien un número máximo de veces.

- **Flujo de datos**

  Esta estrategia se basa en diseñar casos de prueba en cuya ejecución una determinada variable o en general una estructura de datos, toman diferentes valores.

♦ **Caja Negra**

Las pruebas que utilicen el método de Caja Negra se centran en los requerimientos funcionales del software. Permite al ingeniero de software derivar conjuntos de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales de un programa.

La prueba de caja negra no es una alternativa a las pruebas de caja blanca. Más bien se trata de un enfoque complementario, donde se intenta descubrir errores que no pueden ser descubiertos con las pruebas de caja blanca.

Los errores que se pueden encontrar con las pruebas de caja negra están enmarcados en las siguientes categorías:

- Funciones incorrectas o ausentes.

- Errores de interfaz.
- Errores en estructura de datos o en acceso a base de datos.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Para llevar a cabo las pruebas de caja negra, también existen varias estrategias o técnicas, a continuación se muestran las más usadas:

- **Técnica de Partición de Equivalencia**

  Esta técnica divide el campo de entrada en clases de datos: válidos[3] e inválidos[4], y se prueba el funcionamiento del sistema frente a estas clases de datos, que en todo momento el sistema responda según se espera.

- **Técnica del Análisis de Valores Límites**

  Esta técnica prueba la habilidad o capacidad del programa para manejar datos de entrada que se encuentran en los límites aceptables.

- **Técnica de Grafos de Causa-Efecto:**

  Esta técnica permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

≡ **Selección del método de prueba**

Una vez analizados los métodos de pruebas Caja Blanca y Caja Negra, se decidió utilizar para las diferentes pruebas a realizar a T-arenal el método de Caja Negra, por estar enfocado específicamente al diseño de casos de pruebas que ejerciten las funcionalidades de los sistemas a través de una interfaz visual, lo que permitirá verificar que cada funcionalidad esté correctamente implementada.

Como se mostró anteriormente, dentro del método de prueba de Caja Negra existen diferentes técnicas o estrategias, las que se diferencian en la manera de manejar los datos a la hora de probar

[3] Clase de datos válidos: conjunto de datos usados para provocar una respuesta positiva en el programa.
[4] Clase de datos inválidos: conjunto de datos usados para hacer fallar el programa.

determinada funcionalidad. Para las pruebas a T-arenal se hará uso de la técnica de Partición de Equivalencia, en la que se dividen los datos de entrada en clases válidas e inválidas y se comprueba el funcionamiento del sistema frente a estas clases de datos.

1.7 Herramientas utilizadas en las pruebas de software

El proceso de pruebas es repetitivo y necesita de la atención y concentración del probador en todo momento. Desde hace un tiempo se comenzaron a automatizar estas pruebas a través de herramientas que facilitaran, agilizaran y aseguraran este complejo proceso. A continuación se muestran algunas de estas herramientas:

1.7.1 JTest

JTest es una herramienta utilizada en la ejecución de pruebas funcionales en aplicaciones Web. Ofrece varios avances para las industrias de software de alta calidad. Estos avances tecnológicos están concentrados en la realización de pruebas para ayudar a los equipos a verificar de manera automática la funcionalidad de aplicaciones cada vez más complejas, en empresas con sistemas en permanente cambio (J2EE, servicios de SOA/Web), todo esto para generar un incremento en la satisfacción del cliente.

JTest permite reducir el tiempo de entrega del software, así como una disminución del riesgo de generar software defectuoso o con problemas de vulnerabilidad. Para reducir la complejidad de las pruebas, JTest ofrece la generación y ejecución automatizada de los casos de prueba a través de la simulación de un entorno real, posibilita una prueba en tiempo de ejecución que permite la temprana detección de defectos en el código que, de otro modo pasarían inadvertidos hasta etapas avanzadas de desarrollo como el aseguramiento de la calidad.

1.7.2 JMeter

JMeter es una herramienta construida con el lenguaje de programación Java, dentro del proyecto Jakarta, que permite la automatización de pruebas de carga y estrés para aplicaciones web y sus respectivas bases de datos.

Esta herramienta simula la concurrencia de usuarios trabajando en el sistema. Su arquitectura ha evolucionado, no sólo para llevar a cabo pruebas en componentes habilitados en Internet (HTTP), sino además en bases de datos, requisiciones FTP, entre otras posibles pruebas.

JMeter, como herramienta de pruebas, dispone de varios componentes que facilitan la elaboración de escenarios de prueba, con la ventaja de simular para cada uno de ellos miles de usuarios utilizando determinada funcionalidad simultáneamente.

1.7.3 Simulador de conexiones MySQL

Simulador de Conexiones MySQL es una pequeña aplicación implementada en el lenguaje de programación C#, que permite simular un número de conexiones concurrentes y peticiones por cada conexión a una base de datos, y trabajar al mismo tiempo sobre una funcionalidad determinada en forma de consulta.

Dicha aplicación permite conocer el tiempo requerido por el sistema para dar respuesta a todas las peticiones hechas por ***n*** cantidad de usuarios sobre una misma consulta en la base de datos (BD). Por lo que será un buen medidor de la capacidad que tiene el sistema para soportar determinada carga de conexiones sin llegar a perder eficiencia y rapidez durante su funcionamiento.

- **Selección de herramientas de prueba**

La construcción y aplicación de herramientas en el proceso de prueba ha mejorado indudablemente la calidad de las pruebas, y consigo la calidad del producto resultante. Todas estas herramientas son muy efectivas, pero en los últimos tiempos han sido especializadas para aplicaciones web, por lo que no es posible hacer uso de ellas para probar la Plataforma de Cálculo Distribuido T-arenal por ser ésta una aplicación de escritorio (desktop en inglés). Por lo que se utilizó el Simulador de conexiones MySQL.

1.8 Conclusiones

A partir de la investigación y el estudio sobre niveles, técnicas, métodos y herramientas de prueba, se decidió aplicar en las pruebas de Liberación a T-arenal las pruebas Funcionales, de Seguridad y de Carga; haciendo uso del método de Caja Negra, específicamente la técnica de Partición de

Equivalencia, para diseñar los casos de pruebas necesarios para probar todas las funcionalidades del sistema. Y se empleará la herramienta Simulador de Conexiones MySQL para la prueba de carga. Todas estas pruebas se realizarán a nivel de Sistema, probándose el correcto funcionamiento del producto una vez integrados todos sus elementos.

Capítulo 2

Diseño y aplicación de pruebas a T-arenal

2.1 Introducción

En este capítulo se describe la Plataforma de Cálculo Distribuido T-arenal, su composición, requerimientos funcionales, casos de uso, documentación, entre otros aspectos del producto. Se presenta la estrategia que se seguirá para guiar el proceso de pruebas de Liberación y los casos de pruebas diseñados. También se describe el ambiente de pruebas creado para la ejecución de las pruebas a realizar y se detalla cada una de ellas.

2.2 Plataforma de Cálculo Distribuido T-arenal

Un sistema distribuido es un conjunto de computadoras autónomas que aparecen integradas ante los usuarios como una única máquina para resolver determinado problema. Los componentes del sistema no son más que hardware y software que se comunican entre sí a través de una red, preferiblemente canales de alta velocidad.

En el caso de un sistema de cómputo distribuido, el problema a resolver requiere por lo general de grandes cálculos, y se trata de reducir el tiempo de obtención de respuestas. Para ello se hace uso del procesamiento en paralelo, distribuyendo los cómputos de forma transparente para el usuario entre varias computadoras.

T-arenal es un sistema de cómputo distribuido que permite un mayor aprovechamiento de los distintos recursos computacionales disponibles en la red, sin importar su diversidad y heterogeneidad, para realizar cálculos intensos que demandan determinados proyectos bioinformáticos.

El sistema está compuesto por los cuatro módulos descritos a continuación y mostrados en la Figura 2.1 para mejor entendimiento acerca de su arquitectura:

- **Módulo Servidor**

  El módulo servidor almacena el problema a resolver (datos del problema y el algoritmo a usar para su procesamiento) y lo divide en pequeños subproblemas, llamados unidades de trabajo. El principal aspecto del servidor es distribuir las diferentes unidades entre los clientes (computadoras personales, entre otros medios de cómputo) siempre y cuando estos estén autorizados a interactuar con el mismo. Además de desarrollar una lógica de control de unidades pendientes de respuestas y otras que han expirado por algún error ocurrido.

- **Módulo Cliente**

  El módulo cliente está instalado en las máquinas conectadas al servidor mediante una red de área local. El cliente realiza la petición de una unidad de trabajo al servidor, hace el procesamiento de la misma y retorna el resultado al servidor, y vuelve a solicitar otra unidad de trabajo. Múltiples clientes pueden realizar peticiones de trabajo al servidor, quien colecciona el resultado de cada uno de los subproblemas para finalmente construir el resultado del problema original.

- **Módulo Interfaz**

  El módulo Interfaz se utiliza para acceder a todas la funcionalidades en el servidor, tales como: administración de cuentas (adicionar y/o eliminar usuarios y grupos), administración de problemas, así como su ejecución y monitorización, obtener los resultados de las ejecuciones finalizadas, configuración del servidor, entre otras.

- **Módulo de Montajes de aplicaciones**

  El módulo de Montajes de aplicaciones se encarga de implementar aplicaciones que corren sobre la plataforma para distribuir grandes volúmenes de datos, o sea, implementa aplicaciones para correr los problemas que lo necesiten por su demanda de cómputo sobre la plataforma.

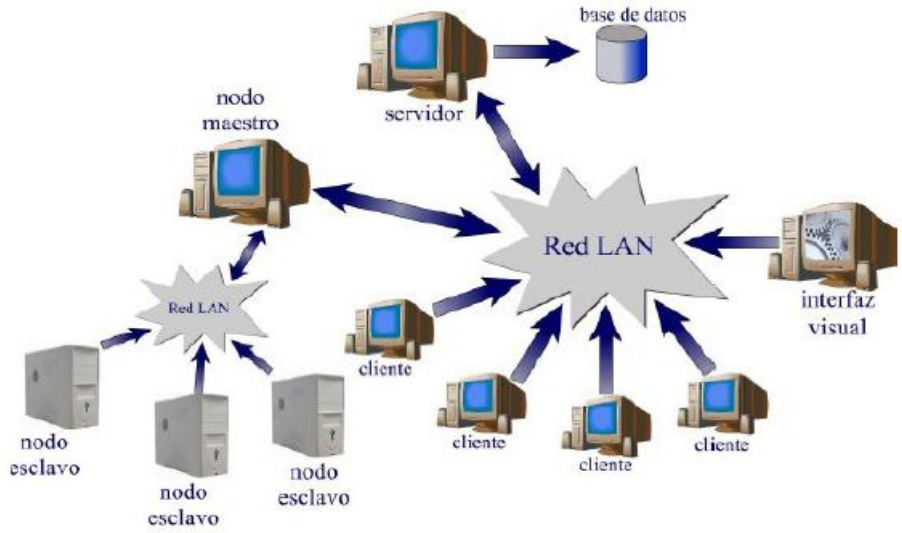

Figura 2.1 Arquitectura del sistema T-arenal

2.2.1 Especificaciones de Hardware y Software

T-arenal es un sistema multiplataforma, o sea, está diseñado para trabajar sobre los sistemas operativos (SO) Windows y Linux. En la Tabla 2.1 se muestran los requerimientos no funcionales mínimos referentes al hardware y al software para cada módulo.

Hardware	Software
SERVIDOR	
- Intel (R) Pentium (R) 4. Velocidad de CPU 3.0 GHz. - 1 GB de RAM.	- SO: Windows XP o GNU/Linux. - Java Runtime Environment (JRE) versión 1.5. - Sistema de Gestión de Base de datos MySQL 5.0.
CLIENTE	
- Intel (R) Pentium (R) 4. Velocidad de CPU 2.4 GHz. - 256 MB de RAM.	- SO: Windows XP o GNU/Linux. - Java Runtime Environment (JRE) versión 1.5.
INTERFAZ	
- Intel (R) Pentium (R) 4. Velocidad de CPU 3.0 GHz.	- SO: Windows XP o GNU/Linux. - Java Runtime Environment (JRE)

- 512 MB de RAM.	versión 1.5.

Tabla 2.1 Especificaciones de Hardware y Software

2.2.2 Requerimientos Funcionales

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer. Estos dependen del tipo de software que se desarrolle, de los posibles usuarios y del enfoque general tomado por la organización al redactarlos.

A continuación se muestran los requisitos funcionales establecidos para T-arenal:

RF 1: Autenticar Usuario.

RF 2: Guardar configuración de acceso al sistema.

RF 3: Gestionar grupos.

RF 3.1: Adicionar un grupo.

RF 3.2: Modificar datos de un grupo.

RF 3.3: Eliminar un grupo.

RF 4: Visualizar datos de un grupo.

RF 5: Mostrar listado de los grupos.

RF 6: Adicionar usuario.

RF 7: Gestionar usuarios.

RF 7.1: Cambiar la contraseña de un usuario.

RF 7.2: Modificar datos de un usuario.

RF 7.3: Habilitar o deshabilitar la cuenta de un usuario.

RF 7.4: Eliminar un usuario.

RF 8: Visualizar datos de un usuario.

RF 9: Mostrar listado de los usuarios.

RF 10: Gestionar rangos IP.

RF 10.1: Adicionar un rango IP.

RF 10.2: Permitir o no un rango IP.

RF 10.3: Eliminar un rango IP.

RF 11: Visualizar datos de un rango IP.

RF 12: Mostrar listado de los rangos IP.

RF 13: Gestionar Clientes.

RF 13.1: Permitir o no un cliente.

RF 13.2: Eliminar un cliente.

RF 14: Mostrar listado de los clientes.

RF 15: Mostrar listado de los clientes autorizados a interactuar con el sistema.

RF 16: Mostrar listado de los clientes no autorizados a interactuar con el sistema.

RF 17: Mostrar listado de los clientes reportados en un tiempo determinado.

RF 18: Mostrar listado de los clientes no reportados en un tiempo determinado.

RF 19: Gestionar Actualizaciones de Clientes.

RF 19.1: Adicionar una actualización.

RF 19.2: Eliminar una actualización.

RF 20: Mostrar listado de las actualizaciones de clientes.

RF 21: Gestionar Problemas.

RF 21.1: Adicionar un problema.

RF 21.2: Eliminar un problema.

RF 22: Mostrar listado de los problemas accedidos por un usuario.

RF 23: Mostrar listado de los problemas administrados por un usuario.

RF 24: Gestionar acceso a problemas.

RF 24.1: Asignar un problema a un usuario.

RF 24.2: Quitar a un usuario el acceso a un problema.

RF 25: Ejecutar un problema.

RF 26: Monitorear ejecuciones.

RF 26.1: Mostrar el estado de una ejecución.

RF 26.2: Mostrar los errores de una ejecución.

RF 27: Gestionar funcionamiento de las ejecuciones.

RF 27.1: Establecer el estado de pausa a una ejecución.

RF 27.2: Quitar del estado de pausa a una ejecución.

RF 28: Cambiar prioridad a una ejecución.

RF 29: Detener una ejecución.

RF 30: Mostrar listado de las ejecuciones.

RF 31: Mostrar listado de las ejecuciones que pertenecen a un usuario.

RF 32: Adicionar una solución.

RF 33: Descargar una solución.

RF 34: Eliminar una solución.
RF 35: Mostrar listado de las soluciones.
RF 36: Mostrar listado de las soluciones que pertenecen a un usuario.
RF 37: Crear una unidad de trabajo.
RF 38: Procesar una unidad de trabajo.
RF 39: Retornar el resultado de una unidad de trabajo.
RF 40: Actualizar versión del cliente del sistema.
RF 41: Mostrar gráfica que represente los clientes por rangos IP.
RF 42: Mostrar gráfica que represente la cantidad de clientes por sistemas operativos.
RF 43: Mostrar gráfica que represente la cantidad de clientes según la capacidad de procesamiento.
RF 44: Mostrar gráfica que represente la cantidad de clientes según la memoria disponible.
RF 45: Realizar chequeo en el servidor del sistema.

2.2.3 Casos de Uso

Para la construcción de T-arenal el equipo de desarrollo decidió diseñar diecinueve casos de uso, mostrados a continuación:

CU 1: Caso de Uso Autenticar Usuario
CU 2: Caso de Uso Gestionar Grupos
CU 3: Caso de Uso Adicionar Usuarios
CU 4: Caso de Uso Gestionar Usuarios
CU 5: Caso de Uso Gestionar Rangos IP
CU 6: Caso de Uso Gestionar Clientes
CU 7: Caso de Uso Gestionar Actualizaciones de Clientes
CU 8: Caso de Uso Gestionar Problemas
CU 9: Caso de Uso Gestionar Acceso a Problemas
CU 10: Caso de Uso Ejecutar Problemas
CU 11: Caso de Uso Monitorear Ejecuciones
CU 12: Caso de Uso Gestionar Funcionamiento de las Ejecuciones
CU 13: Caso de Uso Detener Ejecuciones
CU 14: Caso de Uso Descargar Soluciones

CU 15: Caso de Uso Eliminar Soluciones
CU 16: Caso de Uso Realizar Tarea
CU 17: Caso de Uso Generar Unidades de Trabajo
CU 18: Caso de Uso Mostrar Reportes Gráficos
CU 19: Caso de Uso Realizar Chequeo en el Servidor

2.3 Estrategia de prueba

El objetivo fundamental que se persigue con la realización de las pruebas es la correcta ejecución del software, teniendo en cuenta un conjunto de casos de pruebas, con la primordial intención de detectar fallas o errores que puedan existir en el software, para dar al cliente un mayor nivel de confiabilidad en el sistema desarrollado y por ende, en los procesos que se desarrollarán a partir de la utilización de este producto.

2.3.1 Entorno de prueba

Las pruebas deben ejecutarse en dos arquitecturas de computadoras (hardware y software) diferentes, porque así fue diseñado el sistema, y el ambiente de pruebas debe ser los más semejante posible al ambiente operacional para el cual fue diseñado el producto.

Las arquitecturas se nombraron A y B para diferenciarlas entre sí, y se muestran a continuación en la Tabla 2.2. (Mendoza, 2008).

Arquitectura	Procesador (CPU)	Memoria(RAM)	Sistema Operativo
A	Intel (R) Pentium (R) 4. Velocidad de CPU 2.4 GHz.	256 MB	Windows XP
B	Intel (R) Pentium (R) 4. Velocidad de CPU 3.0 GHz.	512 MB	Ubuntu (Linux)

Tabla 2.2 Especificaciones de arquitecturas

El módulo Interfaz se montará en dos computadoras con arquitecturas como las descritas anteriormente, conectándose a un servidor donde se encontrará montado el módulo Servidor a través

de una red local a 100 Mbps, conjuntamente se conectarán varias computadoras con el módulo Cliente instalado y se pondrá en funcionamiento el sistema.

2.3.2 Proceso de prueba

El proceso de prueba está definido por un conjunto de actividades, las que servirán de guía a todos los implicados en las pruebas:

- Definición de los niveles de prueba.
- Selección de las técnicas y métodos de pruebas para cada nivel.
- Revisión de la documentación referente al sistema.
- Realización de pruebas Exploratorias a la aplicación.
- Diseño de los Casos de Pruebas.
- Realización de pruebas Funcionales al sistema.
- Realización de pruebas de Seguridad al sistema.
- Realización de pruebas de Carga al sistema.
- Documentar los resultados de las pruebas y no conformidades detectadas.
- Análisis de los resultados de las pruebas.
- Evaluación del producto.

Para la ejecución de las pruebas se probará cada uno de los escenarios descritos en los casos de pruebas, haciendo uso de juegos de datos brindados por el equipo de desarrollo, y otros datos que los probadores estimen necesario para cumplir el objetivo de las pruebas.

Las pruebas se realizarán por iteraciones, al finalizar cada una de estas, se le entregará oficialmente al equipo de desarrollo el documento No Conformidades con los fallos o errores encontrados, y tendrán un plazo establecido para dar respuestas a estas no conformidades detectadas. Luego de la revisión de las respuestas a las no conformidades entregadas por el equipo de desarrollo, se procederá con el comienzo de otra iteración.

Tanto los casos de pruebas, como las No Conformidades serán almacenados en un servidor puesto a disposición por el equipo de desarrollo.

2.4 Ejecución de pruebas

Para la realización de las pruebas se planificó conjuntamente con el equipo de desarrollo el tiempo en que se realizarían, recursos necesarios, artefactos, entregables, entre otros aspectos de vital importancia a la hora de planificar y ejecutar las pruebas.

Las pruebas se ejecutaron por dos Probadores, con la presencia en todo momento de un integrante del equipo de desarrollo, quien aseguró el correcto funcionamiento del sistema en el transcurso del proceso.

Cronograma de Trabajo

El cronograma de trabajo contiene las actividades realizadas durante el proceso de prueba realizado a T-arenal; conjuntamente con las actividades se muestra el período en que se realizaron (dado por las fechas de inicio y fin), los responsables y el grado de cumplimiento de estas actividades (dado en porciento).

#	Actividad	Fecha de Inicio - Fin	Responsables	Cumplimiento (%)
1	Reunión de Inicio.	08/12/2008	Grupo Desarrollo Grupo de Prueba	100
2	Capacitación a los probadores sobre el producto.	09/12/2008	Grupo Desarrollo	100
3	Entrega de la Aplicación y la documentación de T-arenal.	09/12/2008	Grupo Desarrollo Grupo de Prueba	100
4	Realización del Plan de Pruebas.	10/12/2008 - 15/12/2008	Yosvany	100
5	Revisión del documento Especificación de Requisitos.	06/01/2009 - 06/01/2009	Yosvany	100
6	Revisión del documento Diccionario de Datos.	07/01/2009 - 07/01/2009	Yosvany	100
7	Revisión del documento Manual de Usuario.	08/01/2009 - 10/01/2009	Yosvany	100
8	Revisión del documento Manual del Desarrollador.	11/01/2009 - 13/01/2009	Yosvany	100
9	Revisión del documento Manual de Instalación.	14/01/2009 - 14/01/2009	Yosvany	100
10	Revisión de las Descripciones de Casos de Uso.	15/01/2009 - 18/01/2009	Arles y Yosvany	100
11	Registro de las no conformidades encontradas en la Documentación.	19/01/2009 - 20/01/2009	Arles y Yosvany	100

12	Corrección de fallas y defectos encontrados (Documentación).	21/01/2008 - 23/01/2009	Grupo Desarrollo	100
13	Revisión de las respuestas a las no conformidades.	24/01/2008 - 24/01/2009	Arles y Yosvany	100
14	Diseño de Casos de Pruebas.	25/01/2009 - 01/02/2009	Arles y Yosvany	100
15	Realización de las Pruebas Exploratorias.	02/02/2009 - 02/02/2009	Arles y Yosvany	100
16	Realización de pruebas Funcionales.	03/02/2009 - 12/02/2009	Arles y Yosvany	100
17	Realización de pruebas de Seguridad.	13/02/2009 - 13/02/2009	Arles y Yosvany	100
18	Registro de las no conformidades encontradas en el Sistema.	14/02/2009 - 14/02/2009	Arles y Yosvany	100
19	Corrección de fallas y defectos encontrados (Sistema).	15/02/2009 - 17/02/2009	Grupo Desarrollo	100
20	Revisión de las respuestas a las no conformidades.	18/02/2009 - 18/02/2009	Arles y Yosvany	100
21	Realización de Pruebas Funcionales (2da iteración).	19/02/2009 - 21/02/2009	Arles y Yosvany	100
22	Registro de las no conformidades encontradas en el Sistema.	22/02/2009 - 22/02/2009	Arles y Yosvany	100
23	Corrección de fallas y defectos encontrados (Sistema).	23/02/2009 - 25/02/2009	Grupo Desarrollo	100
24	Revisión de las respuestas a las no conformidades (2da iteración).	26/02/2009 - 26/02/2009	Arles y Yosvany	100
25	Realización de las pruebas de Carga.	27/02/2009 - 27/02/2009	Arles y Yosvany	100
26	Evaluación del producto.	28/02/2009 - 01/03/2009	Arles y Yosvany	100
27	Liberación del producto.	02/03/2009	Arles y Yosvany	100

Tabla 2.3 Cronograma de Trabajo

2.4.1 Pruebas a la Documentación del sistema

“Los errores en la documentación pueden ser tan destructivos para la aceptación del programa, como los errores en los datos o en el código fuente”. (Pressman, 2001).

La documentación es un buen medidor de la organización, tanto del grupo de desarrollo encargado de la construcción del software, como del producto en sí, y tiene que estar a la altura del software producido ya que es complemento de éste. Una documentación clara y completa es señal de calidad.

“La prueba de documentación se puede enfocar en dos fases. La primera fase, la revisión e inspección, examina el documento para comprobar la claridad editorial. La segunda fase, la prueba en vivo, utiliza la documentación junto al uso del programa real”. (Pressman, 2001).

Dentro de las pruebas realizadas a T-arenal tuvo lugar la prueba de documentación, esta se realizó en dos fases: primero se revisó la documentación desde el punto de vista de redacción, ortografía y concordancia; como segunda fase se revisó su relación con el sistema, verificando que todo lo plasmado en ella estuviera presente en la aplicación y viceversa, que todas las funcionalidades del sistema estuvieran descritas en la documentación.

A continuación se muestran los documentos implicados en la prueba, junto a una breve descripción de cada uno de ellos:

- **Especificación de Requisitos**
  Es uno de los principales documentos, por contener las funcionalidades que debe realizar el sistema, o sea, los requerimientos acordados con el cliente, tanto funcionales, como no funcionales.

- **Descripción de Casos de Uso**
  Existen un total de diecinueve documentos, uno por cada caso de uso definido para el sistema. Cada documento contiene la descripción del caso de uso, los actores implicados, las condiciones de ejecución, la prioridad y el proceso de interacción detallado entre el actor y el sistema.

- **Diccionario de Datos**
  En este documento se registran las palabras empleadas por el equipo de desarrollo que pueden resultar difíciles de entender por personas ajenas a éste, conjuntamente con su significado.

- **Manual de Usuario**
  Este documento describe clara y detalladamente cómo funciona el sistema, por lo que resulta la guía principal para todas aquellas personas que trabajarán con éste, mostrando los pasos definidos para trabajar con la aplicación en cada funcionalidad.

- **Manual del Desarrollador**
  Este documento conforma una guía dedicada específicamente para las personas que trabajarán con la aplicación en el rol de administradores, los que deben tener conocimientos de

programación; en éste se presentan las clases (códigos) con las que se debe trabajar para lograr el objetivo del sistema.

- **Manual de Instalación**
  Es el manual donde se describe, paso por paso, todo el proceso de instalación de cada módulo del programa.

Los defectos encontrados en estas pruebas fueron registrados como no conformidades, para su posterior corrección.

El proceso de revisión de la documentación se planificó para una semana y se realizó en ese mismo tiempo, para ello se utilizó una computadora, la documentación entregada por el equipo de desarrollo y el registro de No Conformidades. Se realizaron dos iteraciones en todo el proceso.

2.4.2 Realización de pruebas Exploratorias

Las pruebas Exploratorias son las primeras que se realizan al sistema por parte del equipo de pruebas, con el fin de encontrar errores superficiales que puedan existir. Por ejemplo, en la navegación a través de las interfaces mostradas, la lógica entre ellas y la correspondencia de los vínculos con sus correspondientes interfaces, por citar algunos.

Por su sencillez no se utilizan métodos para su ejecución, lo que no significa que sean menos importantes o se puedan ignorar. Estas pruebas se desarrollan mediante una interacción exploratoria de los probadores con el sistema, y ellos mismos deciden el tiempo de duración de estas pruebas.

Las pruebas Exploratorias también formaron parte del conjunto de pruebas aplicadas al software T-arenal; por la poca complejidad del grupo de interfaces solamente se dedicó un día a ellas.

2.4.3 Diseño de los Casos de Prueba

“El objetivo del proceso de diseño de casos de prueba es crear un conjunto de casos de pruebas que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requerimientos”. (Sommerville, 2005).

Los casos de prueba se diseñaron a partir de los casos de uso de T-arenal, los que responden a los requisitos del mismo; para ello se utilizó el método Caja Negra, específicamente de la técnica Partición de Equivalencia.

Con la técnica de Partición de Equivalencia se evaluó las clases de equivalencia para una condición de entrada. Las clases se definieron según las siguientes directrices:

1. Si una condición de estrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
2. Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos no válidas.
3. Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida.
4. Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida.

Se diseñaron diecinueve casos de prueba, un caso de prueba por cada caso de uso. En un principio solamente se realizaron los casos de pruebas para los casos de uso críticos del sistema, y posteriormente se integraron los restantes. Para su mejor identificación se le nombró del mismo modo que el caso de uso al cual responde. Estos casos de prueba se refinaron en el transcurso de las pruebas para su mayor consistencia y así aumentar las probabilidades de encontrar errores a la hora de aplicarlos.

El caso de prueba se diseñó sobre una plantilla definida por el laboratorio de calidad de la UCI, a modo estándar, para ser utilizado en todos los proyectos producidos en la misma. Estas plantillas sufren modificaciones con el objetivo de obtener casos de prueba mejores diseñados, para que garanticen detectar la mayor cantidad de errores posibles en los productos.

La plantilla utilizada para estos casos de prueba fue la propuesta a principio del año 2009. Dicha plantilla está compuesta por varias secciones, las que se describirán brevemente a continuación para su mejor entendimiento.

- **Control de versiones:** los casos de pruebas se pueden ir refinando, y en esta tabla se registran las versiones por la que pasa la plantilla.

Fecha	Versión	Descripción	Autor

Tabla 2.4 Control de versiones

- **Descripción General:** en esta sección se describe brevemente el caso de uso correspondiente al caso de prueba.
- **Condiciones de Ejecución:** en esta sección se muestran las condiciones necesarias para la ejecución del caso de uso referente al caso de prueba.
- **Secciones a probar en el caso de uso:** en esta tabla se describe el caso de uso correspondiente al caso de prueba, a través de sus columnas.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central

Tabla 2.5 Secciones a probar en el caso de uso

- **Nombre de la sección**: en esta columna se nombra la sección o las secciones en las que está dividido el caso de uso. Se enumeran de la siguiente manera: SC1, SC2,…, SCX.
- **Escenarios de la sección:** en esta columna se muestran los escenarios posibles dentro de una sección determinada. Se enumeran de la siguiente manera: EC1.1, EC2.1,…, ECX.Y.
- **Descripción de la funcionalidad:** aquí se describe muy brevemente la funcionalidad del escenario.
- **Flujo Central:** esta columna pertenece al flujo central del escenario, descrito detalladamente en el caso de prueba.

- **Descripción de las variables:** en esta tabla se describen las variables a utilizar en el caso de prueba.

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción

Tabla 2.6 Descripción de las variables

- **No:** es el número que identificará la variable.

- **Nombre del campo:** es el nombre de la variable.
- **Clasificación:** especifica el tipo de dato que puede tomar la variable.
- **Puede ser nulo:** especifica si la variable puede ser nula en algún momento.
- **Descripción:** breve descripción de la variable.

- **Secciones:** esta tabla se hace para cada sección, mostrándose de ella las clases válidas e inválidas que se probarán.

Id del escenario	Escenario	Variables		Respuesta del Sistema	Resultado de la Prueba

Tabla 2.7 Secciones

- **Id del escenario:** identificador del escenario que se probará.
- **Escenario:** nombre del escenario que se probará.
- **Variables:** se enumeran las variables que participan en esta sección específica y los valores que toman para formar las clases. Las variables pueden ser: válidas (V), inválidas (I) o no aplica (N/A) para las variables que no participan en determinado escenario.
- **Respuesta del Sistema:** se describe lo que el sistema debe hacer frente a determinada clase (V o I).

- **Registro de defecto y dificultades encontradas:** en esta sección se registran los fallos encontrados específicamente en el caso de uso que se está probando.

La tabla se describirá en el próximo capítulo, en el apéndice referente a la descripción de la plantilla de No Conformidades.

Elemento	No.	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo de Desarrollo

Tabla 2.8 Registro de defectos y dificultades

2.4.4 Pruebas Funcionales

Las pruebas funcionales se realizaron para verificar que el sistema implementa adecuadamente cada uno de los requisitos acordados para el software y que funcionan correctamente.

Haciendo uso de los casos de prueba diseñados, donde se recogen los escenarios correspondientes a los casos de uso del sistema y las clases de datos utilizadas para probar estos escenarios se probó cada funcionalidad, comparando los resultados obtenidos con los resultados esperados, estos últimos también contenidos en los casos de prueba. De esta manera se verificó que el sistema respondía a las necesidades del cliente de la misma manera que se solicitó.

Se revisó también que las interfaces de la aplicación utilizaran correctamente el idioma definido, mantuvieran concordancia entre sus textos y buena ortografía, que los mensajes mostrados fueran claros y concisos. Otro aspecto que se tuvo en cuenta fue la estética y visualización de las interfaces, entre otras características esenciales que debe tener el software o de lo contrario podrían restarle calidad.

Estas pruebas tuvieron un tiempo de duración de 15 días, y para ellas se utilizaron dos computadoras con las arquitecturas requeridas para el producto, los documentos Especificación de Requisitos, Descripción de Casos de Uso y Caso de Prueba. Los errores encontrados fueron documentados en su correspondiente registro de No Conformidades. Se realizaron dos iteraciones en todo el proceso.

2.4.5 Pruebas de Seguridad

Cualquier sistema basado en computadora que maneje información sensible es un posible objetivo para entradas impropias o ilegales a este.

“Las pruebas de seguridad intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán, de hecho, de accesos impropios”.

“Con tiempo y recursos suficientes, una buena prueba de seguridad terminará por acceder al sistema. El papel del diseñador del sistema es hacer que el coste de la entrada ilegal sea mayor que el valor de la informacion obtenida”. (Pressman, 2001).

De modo general, “a través de las pruebas de seguridad se garantiza que la aplicación siga funcionando correctamente frente a cualquier ataque, que custodie los datos que maneja y los proteja frente a manipulación consciente o inconsciente de los usuarios y que su disponibilidad esté garantizada”. (Pérez, 2005).

Las pruebas de Seguridad se realizaron en dos fases y a nivel de usuario, la primera para probar la seguridad que brinda el sistema a la hora de autenticarse, donde se verificó que solo los actores con acceso al sistema están habilitados para accederla; y la segunda para probar la seguridad que brinda el sistema a sus funcionalidades, donde se verificó que un actor solamente tuviera acceso a las funciones y datos que su usuario tiene permitido.

Para la primera fase de la prueba se crearon usuarios en el sistema y se intentó acceder a éste con falsas contraseñas, al mismo tiempo con usuarios no registrados previamente.

Para la segunda fase se identificaron los tipos de usuarios del sistema, las áreas donde pueden acceder y las funciones que pueden realizar cada uno de ellos. Luego se procedió con la ejecución de todas las funciones pasando por los diferentes roles, intentando en todo momento violar los permisos asignados.

El sistema T-arenal cuenta con tres tipos de usuarios, mostrados a continuación:

- **Guest:** es el menor privilegio. Solamente realiza las funciones básicas: cambiar contraseña, acceder a los problemas que tiene permiso, a sus ejecuciones y soluciones.
- **Problem Manager:** realiza las mismas funciones definidas para el usuario anterior; además puede adicionar y administrar problemas al sistema.
- **Administrator:** accede a todas las funcionalidades del sistema.

2.4.6 Pruebas de Carga

Una prueba de carga a un sistema informático se realiza para observar el comportamiento de éste bajo una cantidad determinada de peticiones. La carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Esta prueba puede calcular los tiempos de respuesta de todas las transacciones importantes de la aplicación.

El objetivo fundamental es asegurar que el sistema funciona apropiadamente en circunstancias de carga extrema. Además evaluar características de performance como tiempos de respuesta, tiempos de transacciones y otros elementos sensitivos al tiempo.

Como el software T-arenal es un sistema basado en la arquitectura cliente – servidor, fue significativamente importante saber hasta qué punto puede cargarse de trabajo el sistema sin fallar.

Simulador de conexiones MySQL. Modo de utilización

Para utilizar el Simulador de Conexiones MySQL se introduce primeramente la cadena de conexión relativa a la base de datos, donde se especifica su dirección, puerto de conexión, datos necesarios para realizar la autenticación, la consulta que se desee ejecutar, la cantidad de conexiones que se va a realizar y por último la cantidad de peticiones de la consulta por cada una de estas conexiones. El programa calcula el tiempo que necesita el sistema para dar respuestas a todas las peticiones hechas por cada usuario.

A continuación se describe un juego de datos como ejemplo para usarlo, conjuntamente con la Figura 2.2 donde se muestra la interfaz del simulador para mejor comprensión:

Cadena de conexión	"Dsn= nombre Data source; database= nombre BD; option=0; port=[puerto]; server=[Servidor]; uid=[Usuario]"
Consulta	"INSERT INTO Prueba (nombre) VALUES ('yosvany')"
Conexiones	50
Peticiones	20

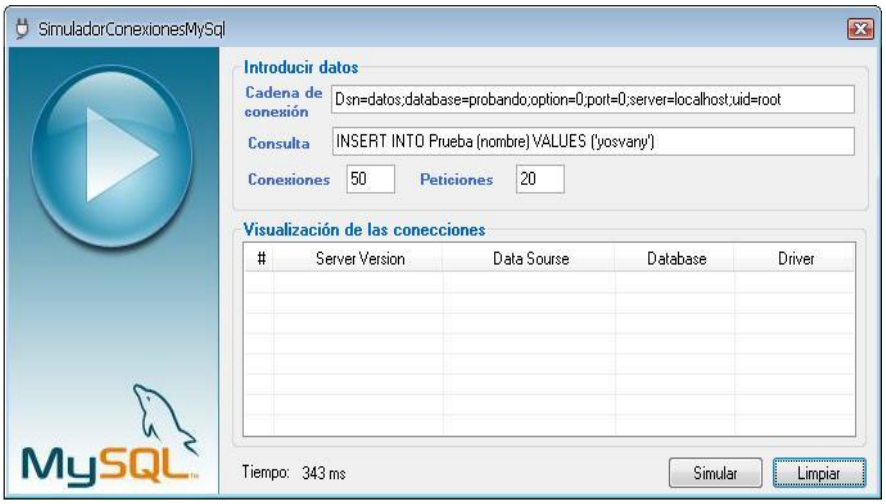

Figura 2.2 Interfaz del Simulador de conexiones MySQL

Coordinado con el equipo de desarrollo de T-arenal se ejecutó esta prueba. Para ello, se creó un escenario de pruebas lo más semejante posible al ambiente donde se desplegará el software una vez entregado al cliente.

Como se especificó en los requerimientos del sistema, el servidor debe ser una computadora Intel (R) Pentium (R) 4, con microprocesador a 3.0 GHz de velocidad y 1 GB de memoria RAM. Para las pruebas no fue posible la obtención de una computadora con idénticas características, pero se realizó sobre una computadora con 512 Mb de memoria RAM (la mitad de lo especificado en los requisitos), lo que aseguraría que si las pruebas resultaban satisfactorias en esta computadora, el sistema funcionaría sin problema alguno en una computadora con mayor cantidad de memoria RAM.

Para la realización de las pruebas de carga, se utilizaron dos consultas de las que se ejecutan con mayor frecuencia en la base de datos del sistema y que responden a las funcionalidades Gestionar Usuarios y Gestionar Problemas. El Simulador de conexiones MySQL simuló varios clientes y peticiones simultáneas sobre la misma consulta. Dichas consultas se muestran en el Anexo 3.

Se realizaron dos iteraciones, la primera para obtener el tiempo de respuesta del sistema al ejecutar las consultas para 50 clientes, porque el entorno para el cual el software fue creado el producto cuenta con aproximadamente 50 computadoras para trabajar como clientes; y la segunda para 150 clientes, para asegurar que el sistema trabajará con igual eficiencia en caso de aumentar en un futuro esa cantidad de computadoras.

2.5 Documentar los resultados. Registro de No Conformidades

“Una no conformidad es el incumplimiento de un requisito”. (ISO, 2000).

Los resultados de las pruebas deben ser bien documentados, de manera que los desarrolladores del software tengan una guía oficial por la cual regirse a la hora de corregir los errores de cada una de las partes del sistema, antes que el proyecto pase a una fase de mayor complejidad donde la corrección de errores se hace más compleja y costosa.

El orden y la organización a la hora de redactar el documento de los resultados juegan un papel importante, tributa a la calidad del trabajo del equipo de pruebas. Es válido resaltar la importancia que tiene el nivel de detalle con que se llene esta plantilla, debe hacerse con extremo cuidado y claridad.

Este registro es entregado al finalizar cada iteración al equipo de desarrollo, y su entendimiento acerca de los defectos detectados depende mucho de la capacidad descriptiva de los probadores.

La plantilla de No Conformidades recoge los errores que son detectados durante las pruebas a la aplicación y la revisión de la documentación del sistema. Se elabora un documento por cada iteración realizada y se controlan a través de versiones según se vayan eliminando los errores, hasta que finalmente se hayan erradicado todos los defectos que posea el elemento que se prueba.

Las no conformidades detectadas durante el proceso de prueba a T-arenal, se entregaron al finalizar cada iteración al equipo de desarrollo, el que tenía un plazo de 72 horas para dar respuesta a estos problemas.

Las respuestas a las no conformidades detectadas se entregaron al equipo de prueba en la misma plantilla de No Conformidades, conjuntamente con la nueva versión del producto con los errores ya corregidos.

2.6 Evaluación del producto

La evaluación del producto es la actividad final de la estrategia de prueba trazada. Como su nombre indica, es una evaluación general emitida por el grupo de probadores acerca del producto, teniendo en cuenta ciertos atributos de calidad que no deben faltar en ningún software.

Para ello se hace uso de una lista de chequeo como mecanismo y función básica para la valoración del producto, donde se encuentran recogidas un conjunto de preguntas, como resultado del trabajo y experiencia de un equipo de especialistas en el tema. Las respuestas a estas preguntas conducirán a la exposición de criterios acerca del artefacto que se está probando.

2.6.1 Listas de Chequeo

“Se entiende por lista de chequeo un listado de preguntas, en forma de cuestionario, que sirve para verificar el grado de cumplimiento de determinas reglas establecidas a priori con un fin determinado". (Bichachi, 2002).

Es un instrumento de medición y evaluación, que consiste básicamente en un formulario de preguntas relacionadas con los atributos de calidad que se están midiendo y con las características del artefacto.

Cada pregunta tiene asociada una evaluación que da una medida del grado de cumplimiento de la pregunta en el artefacto.

El empleo de las listas de chequeo está generalizado a usos muy diversos, que van desde verificar y determinar el potencial de mercados, hasta medir la confiabilidad y seguridad de sistemas informáticos, incluyendo aspectos tales como la evaluación de criterios de usabilidad de un sitio de Internet, por solo citar un ejemplo.

Para la evaluación de T-arenal se utilizó una lista de chequeo (ver Anexo 4), cuyas preguntas están enfocadas directamente a la medición de los atributos: Seguridad, Portabilidad, Usabilidad y Confiabilidad.

A continuación se muestra qué son estos atributos de calidad y qué le aportan a los sistemas:

2.6.1.1 Seguridad

Los sistemas basados en computadoras manejan frecuentemente información muy valiosa para sus usuarios. La protección de esa información contra el uso no autorizado es preocupación esencial de todos los desarrolladores de estos sistemas.

La seguridad indica que un sistema está libre de peligro, daño o riesgo. Se entiende como peligro o daño todo aquello que pueda afectar su funcionamiento directo o los resultados que se obtienen del mismo.

Para que un sistema se pueda definir como seguro debe tener estas tres características:

- **Integridad:** La información sólo puede ser modificada por quien está autorizado y de manera controlada.
- **Confidencialidad:** La información es accedida solamente por personas autorizadas.
- **Disponibilidad:** Debe estar disponible cuando se necesita.

2.6.1.2 Portabilidad

La portabilidad es uno de los conceptos clave en la programación de alto nivel. Es la característica que posee un software para ejecutarse en diferentes plataformas; el código fuente del software es capaz de reutilizarse en vez de crearse un nuevo código cuando el software pasa de una plataforma a otra.

A mayor portabilidad menor es la dependencia del software con respecto a la plataforma.

2.6.1.3 Usabilidad

"La usabilidad es la disciplina que estudia la forma de diseñar programas para que los usuarios puedan interactuar con ellos de la forma más fácil, cómoda e intuitiva posible". (Montero, 2004).

La usabilidad debería ser considerada en todo momento, desde el mismo comienzo del proceso de desarrollo hasta las últimas acciones antes de hacer el sistema, producto o servicio disponible al público. Ésta se basa en tres principios básicos:

- **Facilidad de Aprendizaje:** facilidad con la que nuevos usuarios pueden tener una interacción efectiva. Está relacionada con la sintetización, familiaridad, la generalización de los conocimientos previos y la consistencia.
- **Flexibilidad:** variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información. También abarca la posibilidad de diálogo, la multiplicidad de vías para realizar la tarea, similitud con tareas anteriores y la optimización entre el usuario y el sistema.
- **Robustez:** nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos. Está relacionada con la capacidad de observación del usuario, de recuperación de información y de ajuste de la tarea al usuario.

A continuación se muestran algunos de los beneficios de la usabilidad:

- Reducción de los costes de aprendizaje.
- Disminución de los costes de asistencia y ayuda al usuario.
- Optimización de los costes de diseño, rediseño y mantenimiento de los sitios.
- Aumento de la tasa de conversión de visitantes a clientes del sitio web.
- Mejora la imagen y el prestigio del sitio web.

- Mejora la calidad de vida de los usuarios del sitio, ya que reduce su stress, incrementa la satisfacción y la productividad.

2.6.1.4 Confiabilidad

La confiabilidad de un sistema informático es una propiedad igual a su fidelidad. La fidelidad esencialmente significa el grado de confianza del usuario en que el sistema operará tal y como se espera de él y que no fallará al utilizarlo normalmente.

Esta propiedad no se puede expresar numéricamente, sino que se utilizan términos relativos como: **no confiable**, **poco confiable**, **muy confiable**, **ultra confiable** para reflejar los grados de confianza que se puede tener en un sistema.

Dentro de las dimensiones de la confiabilidad existen cuatro principales, tal y como se muestra a continuación en la Figura 2.2:

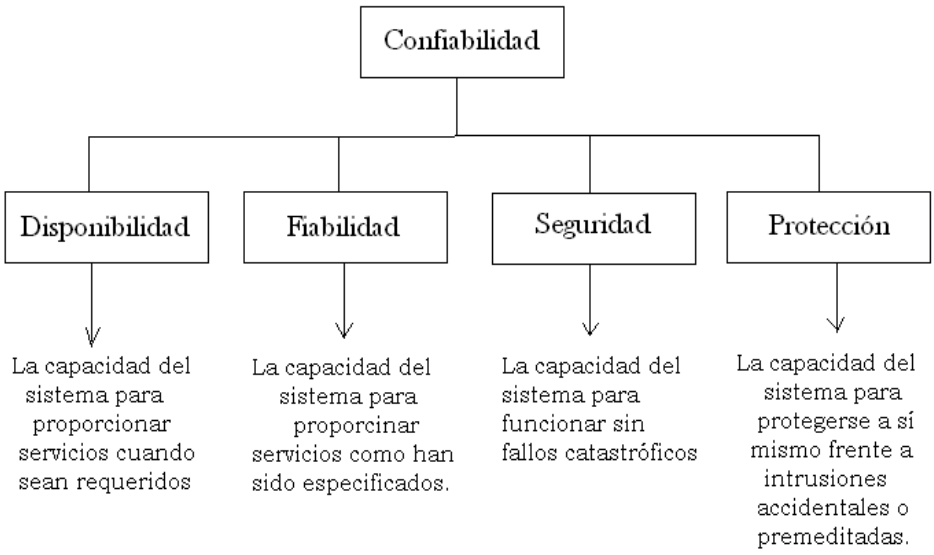

Figura 2.3 Dimensiones de la confiabilidad

Las propiedades de confiabilidad mostradas anteriormente están interrelacionadas. El funcionamiento de un sistema seguro depende normalmente de que el sistema esté disponible y su funcionamiento sea fiable.

2.7 Conclusiones

En el capítulo se describió detalladamente todo el proceso de pruebas realizado a la Plataforma de Cálculo Distribuido T-arenal. Se planificó y desarrollo la estrategia de pruebas, seguida por el proceso de ejecución de cada prueba. También se diseñaron los casos de prueba necesarios para probar las funcionalidades de T-arenal y se presentó la plantilla utilizada para la realización de estos, siguiendo la premisa de que un buen diseño de casos de prueba es aquel que asegurará encontrar la mayor cantidad de posibles defectos existentes en cualquier producto software.

El registro de los defectos encontrados a lo largo del proceso de pruebas en la plantilla de No Conformidades establecida en la UCI, también fue parte de la estrategia de pruebas trazada. Estos errores se expondrán y analizarán en el próximo capítulo. Finalmente se presentó la lista de chequeo que se utilizará en la evaluación final del producto, que contiene un conjunto de preguntas necesarias para chequear la seguridad, portabilidad, usabilidad y confiabilidad en el sistema.

Capítulo 3
Análisis de los Resultados

3.1 Introducción

Para que un proceso de pruebas tenga éxito se requiere de un análisis final de los resultados arrojados, es decir, la evaluación del producto que se está probando de acuerdo a todos los defectos y fallos del sistema encontrados a lo largo del proceso.

En este capítulo se analizarán las no conformidades detectadas en las distintas pruebas realizadas a la Plataforma de Cálculo Distribuido T-arenal, como son: las pruebas a la documentación y al sistema (Exploratorias, Funcionales, de Seguridad y de Carga). Por último se hace una valoración del producto general tomando como base los parámetros de calidad siguientes: seguridad, portabilidad, usabilidad y confiabilidad, utilizando para ello una lista de chequeo.

3.2 Descripción de la plantilla de No Conformidades

La plantilla de No Conformidades constituye un registro de los defectos y fallos encontrados en el transcurso de las pruebas, cuyo principal objetivo es verificar en un futuro que estos errores fueron erradicados en posteriores iteraciones.

Dentro de la plantilla de caso de prueba, hay una sección dedicada a documentar los errores encontrados en la funcionalidad que se está probando. Pero todos los errores encontrados en las diferentes funcionalidades son registrados en la plantilla de No Conformidades, la que se entrega oficialmente a los desarrolladores.

Dicha plantilla está formada por las siguientes secciones:

- **Control de versiones:** en esta sección se registran las versiones por la que pasa la plantilla de no conformidades.

- **Descripción General:** en esta sección se describen los aspectos generales a tener en cuenta a la hora de realizar el diseño de las pruebas, incidencias en el momento de su desarrollo y otros aspectos relevantes.
- **Elementos probados:** en esta sección se enumeran todos los elementos probados.
- **Elementos no probados y causas:** en esta sección, como lo indica su nombre, se enumeran los elementos que no se probaron y las causas.
- **Registro de defectos y dificultades encontradas:** en esta sección se registran las no conformidades que se encontraron durante las pruebas.
- **Anexos:** esta sección es utilizada para anexar elementos que ayuden a la comprensión de los defectos encontrados.

3.2.1 Registro de defectos y dificultades

La siguiente tabla es la sección más importante dentro de la plantilla de No Conformidades, porque es precisamente donde se recogen los defectos encontrados durante el proceso de pruebas. Por este motivo se decidió describirla detalladamente.

Elemento	No.	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equipo Desarrollo

Tabla 3.1 Registro de defectos y dificultades encontradas

- **Elemento:** especifica el nombre del elemento que se está probando.
- **No.:** especifica el número de la no conformidad correspondiente, es el identificador de ésta.
- **No conformidad:** descripción clara y detallada de la no conformidad.
- **Aspecto correspondiente:** especifica el lugar donde ocurrió la no conformidad. Debe estar bien especificado y puede hacerse uso de imágenes para mostrar exactamente el error y donde ocurrió.
- **Etapa de detección:** especifica en que etapa de las pruebas se descubrió el error.
- **Clasificación:** especifica el tipo de no conformidad.
- **Estado NC:** especifica el estado en que se encuentra la no conformidad detectada, una vez entregadas al grupo de probadores las respuestas a todas las no conformidades. Puede estar: respondida (RA) cuando el equipo de desarrollo dio respuesta a la no conformidad, pendiente (PD)

cuando la no conformidad aún no ha sido respondida, o no procede (NP) cuando el equipo de desarrollo no considera la no conformidad como un problema a resolver.

- **Respuesta del equipo de desarrollo:** especifica la respuesta a cada una de las no conformidades por parte del equipo de desarrollo.

Las no conformidades se clasificaron en Significativas, No Significativas y Recomendaciones, para priorizar los errores encontrados, según la gravedad de estos. Es válido aclarar que el hecho de ser significativas o no, no quiere decir que se le dé menos o más importancia, ambos constituyen errores que hay que reparar.

- **No Significativas:** son los errores detectados que afectan solamente la funcionalidad que se está probando.
- **Significativas:** son aquellos errores que pueden afectar más allá de la funcionalidad que se está probando.
- **Recomendaciones:** son sugerencias dadas por los probadores al equipo de desarrollo para mejorar funcionalidades o para evitar un posible error en el futuro.

3.3 Análisis de los resultados de las pruebas a T-arenal

En el proceso de pruebas es fundamental el registro de la documentación pertinente. Los resultados obtenidos en cada iteración de pruebas, en términos de cantidad de no conformidades por tipo, deben ser registrados y entregados al equipo de desarrollo para su posterior corrección.

Para mejor organización y comprensión de los resultados de las pruebas realizadas a T-arenal, se dividieron en cinco apéndices, uno para los resultados obtenidos a partir de las pruebas a la documentación y otro para los resultados de las pruebas Exploratorias, Funcionales, de Seguridad y de Carga respectivamente.

3.3.1 Pruebas a la Documentación. Resultados

Al revisar la documentación, primeramente, se debe verificar que la misma se encuentre en su totalidad para comenzar con el proceso y tener en cuenta algunos aspectos significativos, tales como son: redacción, ortografía, estructura de la documentación, entre otros.

Los defectos encontrados en cualquier documento deben corregirse por muy insignificantes que puedan parecer, la documentación de un sistema es el reflejo del propio sistema y debe estar a la altura del mismo.

Es importante también recordar siempre que algunos documentos son escritos específicamente para personas que probablemente no tengan conocimientos avanzados de técnicas computacionales, como es el caso de los manuales, por lo que tienen que estar redactados de la forma más entendible y clara posible. Por esta razón los probadores deben ponerse en todo momento en el lugar de estas personas.

La Tabla 3.2 mostrada a continuación, resume la cantidad de no conformidades encontradas en cada uno de los documentos revisados en las dos etapas o iteraciones que se realizaron las pruebas, clasificadas en significativas y no significativas. También se muestra la cantidad de recomendaciones hechas al equipo de desarrollo con el objetivo de entregar al cliente un producto con mayor calidad y con vista a la mejora profesional de los desarrolladores para próximos trabajos.

Documentos	Iteración	No Conformidades	Significativas	No Significativas	Recomendaciones
Especificación de requisitos	**1**	1	1	0	1
	2	0	0	0	0
Manual de usuario	**1**	2	2	0	2
	2	1	1	0	1
Manual del desarrollador	**1**	3	2	1	0
	2	1	0	1	0
Manual de instalación	**1**	0	0	0	1
	2	0	0	0	0
Descripciones de caso de uso	**1**	0	0	0	0

	2	0	0	0	0
Diccionario de datos.	**1**	1	1	0	0
	2	0	0	0	0
Total		**9**	**7**	**2**	**5**

Tabla 3.2 Registro de defectos encontrados en la documentación

De las nueve no conformidades encontradas en la documentación: cuatro fueron faltas de concordancia en la redacción, tres por errores ortográficos cometidos y las restantes por violaciones de estándares definidos por los propios documentadores del equipo de desarrollo para la estructuración de los documentos. El documento donde se detectaron más defectos fue en el Manual del Desarrollador, documento que se le entrega al cliente como ayuda para interactuar con el sistema, por lo que debe estar totalmente libre de defectos. Con la eliminación de los errores encontrados se liberó la documentación, quedando libre de defectos y errores, lista para entregar al cliente.

El registro de no conformidades donde se muestran los defectos encontrados a lo largo del proceso de pruebas a la Documentación se muestra en el Anexo 1.

3.3.2 Pruebas Exploratorias. Resultados

Las pruebas Exploratorias, como se especificó en capítulos anteriores, constituyen la primera iteración entre el grupo de probadores y el sistema, buscando fallas superficiales que puedan existir en las interfaces de la aplicación. T-arenal cuenta con un módulo Interfaz que proporciona el conjunto de interfaces de usuario, a través de las cuales el usuario interactúa con el sistema. Dichas interfaces son bastante sencillas y entendibles, por lo que solamente se dedicó un día a estas pruebas.

Los resultados de estas pruebas fueron positivos, no se encontró ningún problema en los vínculos y lógica entre las interfaces, mostrándose únicamente la información que realmente necesita el usuario en todo momento. Los textos mostrados en las interfaces son coherentes y correctamente redactados.

3.3.3 Pruebas Funcionales. Resultados

Las pruebas funcionales al sistema se realizaron en dos iteraciones y se utilizaron los diecinueve casos de pruebas diseñados, para verificar que las funcionalidades implementadas fueron las acordadas con el cliente y que respondían correctamente a sus necesidades.

El resumen de los defectos encontrados como resultado de estas pruebas, se muestra en la siguiente tabla:

Aplicación	Iteración	No Conformidades	Significativas	No Significativas	Recomendaciones
Interfaz de Usuario	**1**	10	7	3	3
	2	0	0	0	5
Total		**10**	**7**	**3**	**8**

Tabla 3.3 Registro de defectos encontrados en las pruebas Funcionales

Del total de no conformidades encontradas, las clasificadas como significativas están relacionadas con problemas de funcionalidad del sistema, que en muchas ocasiones no coincidían exactamente con las descripciones de los casos de uso a los que responden. Mientras que los defectos clasificados como no significativos están relacionados con problemas principalmente con el idioma en el que se desarrolló el sistema: el inglés, como son textos utilizados en las ventanas y mensajes mal redactados.

Las recomendaciones realizadas al equipo de desarrollo perseguían como objetivo proponer mejoras del sistema o prevenir la ocurrencia de fallas en un futuro. Tal y como se mostró en la Tabla 3.3, se hicieron un total de ocho recomendaciones.

Como se explicó anteriormente, una vez que los probadores entregaban al equipo de desarrollo los defectos encontrados en una iteración, estos últimos contaban con 72 horas para resolverlos. El equipo de desarrollo solucionó cuatro no conformidades; las seis restantes no procedieron.

Una no conformidad que no procede se refiere a un defecto identificado por los probadores y que no es considerado por los desarrolladores como un error, por lo que deben brindar una explicación acerca de por qué no coinciden con la no conformidad. Esto suele suceder por varios motivos; en el documento:

Plantilla de NC T-arenal, se encuentran plasmadas las explicaciones acerca de cada una de las no conformidades que no procedieron.

El equipo de desarrollo refirió que, muchas de las recomendaciones hechas en el proceso de prueba se tomarán en cuenta para la próxima versión del producto. Con la erradicación de los errores detectados a lo largo del proceso de prueba se liberó T-arenal, obteniendo un producto listo para satisfacer las necesidades del cliente.

El registro de no conformidades donde se muestran los defectos encontrados a lo largo del proceso de pruebas Funcionales, se muestra en el Anexo 2.

3.3.4 Pruebas de Seguridad. Resultados

Para la realización de las pruebas de Seguridad a T-arenal, primeramente, se identificaron los distintos tipos usuarios que están definidos en el sistema y sus respectivos niveles de acceso o privilegios definidos; luego se creó un usuario de cada tipo y se procedió con las pruebas.

Las primeras pruebas fueron intentos de violación de la seguridad del sistema, tratando de acceder a éste con usuarios no existentes o utilizando usuarios reales con contraseñas erróneas. Estas pruebas de acceso arrojaron resultados favorables al sistema, ya que no se logró acceder con usuarios no registrados previamente ni con falsas contraseñas, demostrando que T-arenal cuenta con un mecanismo de autenticación seguro, garantizando que solamente trabajen con el programa personas autorizadas previamente.

Luego de acceder al sistema con un usuario específico comenzó la segunda fase de la prueba, donde se intentó acceder a funciones no permitidas para ese determinado usuario, como son: creación o eliminación de usuarios, modificación de los permisos de usuarios existentes, inicialización o terminación de problemas, entre muchas otras funciones de las que depende el correcto funcionamiento del sistema. Este mismo procedimiento se ejecutó una y otra vez con los tres tipos de usuarios con que cuenta la aplicación.

Los resultados de las pruebas fueron los esperados, los usuarios registrados en el sistema T-arenal tienen acceso solamente a las funcionalidades predefinidas por los administradores del sistema, impidiendo en todo momento que se lleven a cabo funcionalidades por personas no autorizadas.

Estas pruebas comprobaron la seguridad dentro de la aplicación, es decir, que solamente pudieran acceder al sistema usuarios registrados y que cada uno de ellos tuviera acceso solamente a la información que le corresponde de acuerdo al rol que desempeña dentro del sistema. El tiempo de duración de estas pruebas fue un día, exactamente el planificado inicialmente.

Se puede afirmar que T-arenal es un sistema que, aparte de cubrir totalmente las funcionalidades definidas por el cliente, brinda suficiente protección a la información que maneja, la protege frente a manipulación consciente o inconsciente de los usuarios, y su disponibilidad está garantizada en todo momento.

3.3.5 Pruebas de Carga. Resultados

Como se especificó en el capítulo anterior, para las pruebas de carga se utilizaron dos consultas correspondientes a las funcionalidades Gestionar Usuarios y Gestionar Problemas; y se varió la cantidad de conexiones simultáneas según los requerimientos del cliente y una cantidad determinada por los probadores para conocer el comportamiento del sistema en caso de aumentar la cantidad de clientes del sistema.

Los resultados arrojados se muestran en la Tabla 3.4, donde se registraron los datos utilizados, y el tiempo que se tomó el sistema para dar respuesta a todos los clientes simulados. Los tiempos de respuestas se expresaron en milisegundos (ms), donde 1 segundo equivale a 1000 milisegundos.

Iteraciones	Consultas	Conexiones	Peticiones	Tiempo de respuestas (ms)
Primera	1	50	1	**187**
		50	50	**625**
	2	50	1	**203**
		50	50	**640**
Segunda	1	150	1	**296**
		150	50	**656**

	2	150	1	**343**
		150	50	**671**

Tabla 3.4 Registro de tiempos de respuestas

En la figura se puede apreciar la cantidad de conexiones simultáneas (clientes) y la cantidad de peticiones que realizaban éstos sobre cada una de las consultas utilizadas en la prueba (Ver Anexo 3). La diferencia entre los colores de las filas está dada precisamente por estas dos consultas, identificando al color azul la consulta 1 y al color blanco la consulta 2.

Los tiempos de respuestas obtenidos como resultado, en todos los casos, son tiempos de espera admisibles para cualquier persona que interactué con una computadora, tiempos que no llegan a 1 segundo. Por lo que se puede afirmar que, el software T-arenal no perderá calidad, ni eficiencia de trabajo en caso de aumentar la cantidad de computadoras trabajando como clientes del sistema.

A continuación se muestran datos obtenidos de una prueba realizada por el propio grupo de desarrollo a T-arenal. La prueba consistió en la ejecución distribuida de un problema biológico llamado DOCK, con el objetivo de disminuir, a través de la distribución de sus cálculos, el tiempo de procesamiento necesario para dar solución a éste. Inicialmente la ejecución comenzó con 1 computadora como cliente y luego se incrementó gradualmente este número, observando el comportamiento del sistema mientras se aumentaron las peticiones concurrentes sobre sus funcionalidades.

“El DOCK es un programa para realizar predicción computacional del modo de unión de dos moléculas, también conocido como **docking molecular**. En este proceso se trata de identificar cuáles moléculas (ligandos) de una base de datos tendrían la capacidad de unirse al sitio activo de la proteína diana. El problema fundamental que se enfrenta consiste en determinar todas las conformaciones que puede tomar el ligando dentro del sitio activo, demandando grandes cómputos para el proceso”. (Mendoza, 2008).

Los resultados de la prueba se muestran en la Tabla 3.5, que contiene la cantidad de computadoras (con sistemas operativos Windows y Linux) utilizadas en el transcurso de la prueba y el tiempo que se tomó el sistema (expresado en segundos) para solucionar el problema con estas computadoras.

PCs Windows	PCs Linux	Total PCs	Tiempo (seg)
0	1	**1**	159585
3	2	**5**	52890
11	4	**15**	24012
40	0	**40**	6991
34	22	**56**	6159
68	10	**78**	5599
77	17	**94**	5152
89	18	**107**	5268
98	20	**118**	5603

Tabla 3.5 Resultados obtenidos a partir de la ejecución del DOCK

Como se puedo observar, a medida que se aumentó el número de computadoras haciendo peticiones concurrente sobre las funcionalidades del sistema, el tiempo requerido para dar respuesta al DOCK disminuyó notablemente y el sistema se mantuvo trabajando correctamente en todo momento; demostrando que el aumento de la cantidad de clientes en el entorno operacional para el cual fue desarrollado el sistema no perjudicará su correcto funcionamiento.

3.4 Evaluación de T-arenal

La evaluación de T-arenal se realizó de dos maneras diferentes; primeramente de forma cuantitativa, donde se obtuvo un resultado numérico de cada atributo de calidad utilizado en la evaluación y a partir de estos se expone un resultado cualitativo de los mismos.

La lista de chequeo usada para la evaluación (Ver Anexo 4) cuenta con un conjunto de preguntas relacionadas a cada atributo de calidad, a estas se le asignaron un peso según su importancia y un valor según el cumplimiento de la pregunta en el sistema.

- Según su importancia: los signos **!!** para las de mayor peso y **!** para las de menor peso.

- Según su valor: 0 - propiedad no disponible, 2 - propiedad parcialmente disponible, 4 - propiedad disponible y 5 - propiedad muy bien implementada.

Una vez aplicadas las preguntas, se sumaron los valores asignados a cada pregunta y se dividió por la cantidad total de estas; así se obtuvo un valor general para cada atributo evaluado. La Figura 3.1 mostrada a continuación, es la gráfica generada a partir de dichos valores.

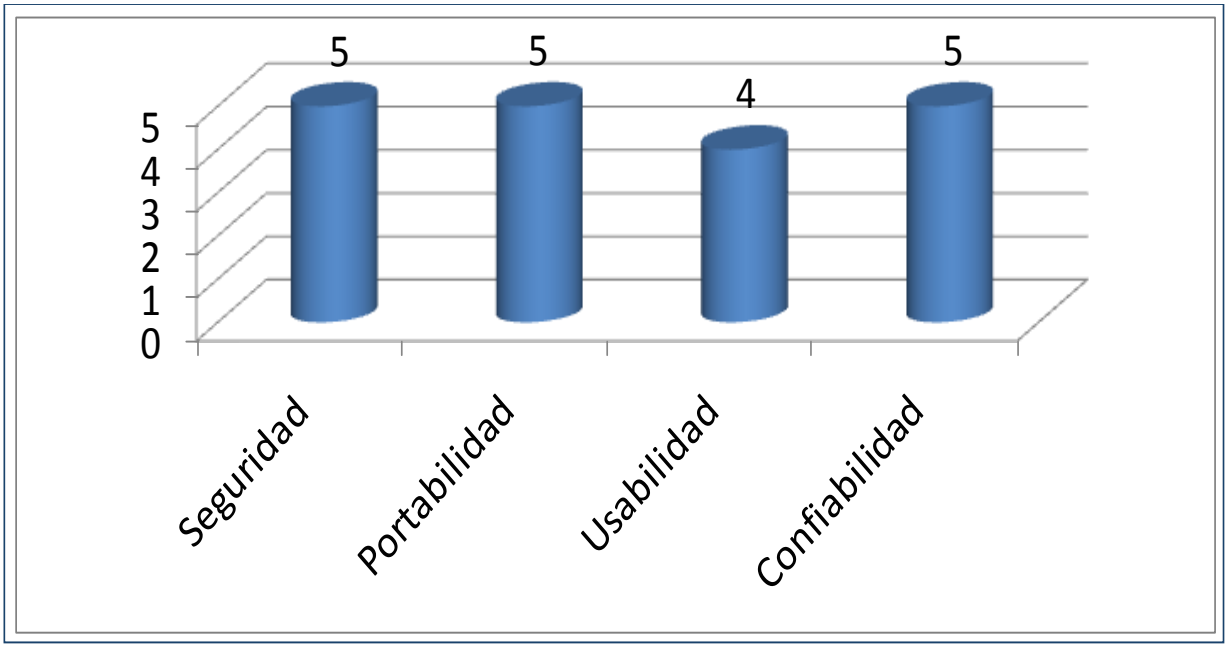

Figura 3.1 Resultados por atributo de calidad

A partir de los resultados cuantitativos obtenidos una vez aplicada la lista de chequeo, T-arenal puede calificarse como un producto:

- **Muy Seguro**

Por ser un sistema que incorpora mecanismos de protección para que solamente accedan a la aplicación las personas previamente autorizadas y éstas pueden realizar únicamente las funcionalidades definidas para su rol. Toda la información que manipula está protegida contra el uso no autorizado y tiene implementada técnicas para la encriptación de la información que se intercambia entre el servidor y los clientes. De manera general el sistema garantiza su funcionamiento frente a cualquier intento de ataque.

- **Muy Portable**

Por ser un sistema multiplataforma, garantizando un correcto funcionamiento independientemente del ambiente de software o hardware utilizado. Por esta razón permite incorporar computadoras al sistema distribuido sin tener en cuenta el sistema operativo ni el hardware con que cuenta.

- **Usable**

Por ser un sistema sencillo de usar por cualquier persona, sin necesidad de tener avanzados conocimientos sobre sistemas informáticos. Cuenta con suficiente documentación como ayuda para todos los que interactúen con él, pero aumentaría su facilidad de uso si se le incorporase un mecanismo de ayuda a la aplicación; también se le podría implementar mecanismos para deshacer y rehacer las acciones que se llevan a cabo durante el trabajo con el sistema e incrementar las vías o caminos existentes para realizar las funcionalidades.

Estos son algunos de los mecanismos que hacen a un sistema muy usable; atributo que se debe tener en cuenta desde los primeros momentos que se comienza a construir un software.

- **Muy Confiable**

Por ser un sistema que responde siempre tal y como lo espera el usuario; teniendo incorporado mecanismos para recuperarse ante cualquier falla, lo que le permite resolver siempre el problema en el que se trabaja. T-arenal brinda gran disponibilidad, lo que garantiza al usuario poder contar con los servicios de este sistema en cualquier momento que lo necesite.

3.5 Conclusiones

El proceso de pruebas a cualquier software se realiza a través de iteraciones, donde, a medida que se procede con una nueva iteración deben haberse erradicado los defectos encontrados en la anterior, para garantizar que al final del proceso el producto quedará libre de la mayor cantidad de errores posible, y a la vez, listo para entregar al cliente.

Luego de haberse realizado un profundo análisis acerca de los resultados de cada una de las pruebas que se ejecutaron como parte del proceso de liberación de la Plataforma de Cálculo Distribuida T-arenal, se puede considerar este producto como: muy seguro, muy portable, usable y muy confiable.

T-arenal ha quedado libre de todos los defectos encontrados en el transcurso de las pruebas, y cumple eficientemente cada una de las funcionalidades acordadas con el cliente; por lo que se puede asegurar que cubrirá las expectativas de éste una vez desplegado en el entorno para el cual fue construido.

CONCLUSIONES

- Se realizaron las pruebas a la documentación perteneciente a T-arenal, para garantizar la entrega al cliente de una documentación libre de errores, a la altura del producto en sí.
- Se diseñaron diecinueve casos de prueba, respondiendo a la cantidad de casos de uso del sistema, donde se recogieron todos los escenarios descritos en estos últimos; lo que permitió probar cada una de las funcionalidades acordadas con el cliente en forma de requerimientos.
- Se realizaron pruebas Exploratorias, Funcionales, de Seguridad y de Carga sobre el sistema, una vez integrados todos los elementos de éste; garantizando que el sistema tiene implementado todas las funcionalidades requeridas, que protege la información que manipula, y que responderá correctamente a las solicitudes de todas las computadoras que trabajarán como clientes del sistema distribuido, sin llegar a colapsar.
- Se documentaron los resultados arrojados por las distintas pruebas, lo que permitió al equipo de desarrollo reparar las fallas encontradas, y hacer mejoras al sistema para posteriores versiones.

RECOMENDACIONES

- Insertar un equipo de aseguradores de calidad a los grupos de desarrollo que aún no cuentan con uno, para que se realicen las pruebas necesarias desde los primeros momentos que se comienza a desarrollar el software y durante todo el ciclo de vida, tal y como lo establecen las metodologías de desarrollo.
- Realizar talleres, conferencias y actividades de intercambio de conocimientos en los grupos de calidad acerca de normas de calidad, técnicas y herramientas de pruebas existentes, con vista a la mejor preparación de éstos, y así asegurar mayor calidad en los productos de la UCI.

REFERENCIAS BIBLIOGRÁFICAS

Bichachi, D. S. (2002). *Instituto Internacional de Estudio y Formación sobre Gobierno y Sociedad.* Recuperado el 20 de febrero de 2009, de http://www.salvador.edu.ar/vrid/iiefgs/docs_investigac.htm: http://www.salvador.edu.ar/vrid/iiefgs/tr_check_list.pdf

Cosín, J. D. (2007). *Técnicas cuantitativas para la gestión en la ingeniería del software.* Netbiblo.

IEEE. (1990). *Computer Dictionary. Computer Society.*

ISO. (2000). *Patente nº 00.012.10.*

Mendoza, L. A. (2008). *Sistema de cómputo distribuido aplicado a la Bioinformática. Tesis en opción al grado de Máster en Bioinformática.* Ciudad de la Habana.

Montero, Y. H. (noviembre de 2004). *No solo Usabilidad.* Recuperado el 10 de abril de 2009, de http://www.nosolousabilidad.com/articulos/introduccion_usabilidad.htm

Pérez, P. G. (2005). *Principios básicos del desarrollo seguro.* División de Seguridad Lógica de Germinus.

Pressman, R. S. (2001). *Ingeniería del Software. Un enfoque práctico. Quinta Edicion.*

Sommerville, I. (2005). *Ingeniería del software. Séptima Edición.* Madrid: Pearson Educación S.A.

UCI. (noviembre de 2008). *La producción en la UCI*. Recuperado el enero de 2009, de Portal UCI - Universidad de la Universidad de las Ciencias Informáticas: http://www.uci.cu/?q=node/46

BIBLIOGRAFÍA

Anna C. Grimán, M. P. (2005). *Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales*. Caracas, Venezuela: Departamento de Procesos y Sistemas. Universidad Simón Bolívar.

Aspiazu, G. C. (2002). *Ingeniería del Software. Principios y Conceptos.* La Paz. Bolivia.

Berzosa, L. R. (2008). Evitando el síndrome de "caja negra" en proyectos de desarrollo software externalizados. (pág. 32). Madrid. España: Expo:QA.

Bichachi, D. S. (2002). *Instituto Internacional de Estudio y Formación sobre Gobierno y Sociedad.* Recuperado el 20 de febrero de 2009, de http://www.salvador.edu.ar/vrid/iiefgs/docs_investigac.htm: http://www.salvador.edu.ar/vrid/iiefgs/tr_check_list.pdf

Cosín, J. D. (2007). *Técnicas cuantitativas para la gestión en la ingeniería del software.* Netbiblo.

Cueva, J. M. (octubre de 2000). *Calidad del Software* . España.

Díaz, J. G. *Introducción al Proceso de Pruebas.* Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla., Sevilla.

EPF Wiki. (s.f.). *OpenUp*. Recuperado el marzo de 2009, de http://epf.eclipse.org/wikis/openup/index.htm

Expo:QA. (2005). *Jornadas Profesionales de Calidad y Testing de Software* . Recuperado el enero de 2009, de http://www.expoqa.com/

García, J. (2003). *Prácticas y métodos para mejorar el desarrollo de Proyectos de Software*. Recuperado el febrero de 2009, de Ingeniero Software: http://www.ingenierosoftware.com/

IEEE. (1990). *Computer Dictionary. Computer Society.*

ISO. (2000). *Patente nº 00.012.10.*

L. S., & Y. P. (2007). *Diseño y aplicación de pruebas al producto Registro Cubano de Discapacitados. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* Ciudad de la Habana. Cuba.

Mendoza, L. A. (2008). *Sistema de cómputo distribuido aplicado a la Bioinformática. Tesis en opción al grado de Máster en Bioinformática.* Ciudad de la Habana.

Montero, Y. H. (noviembre de 2004). *No solo Usabilidad.* Recuperado el 10 de abril de 2009, de http://www.nosolousabilidad.com/articulos/introduccion_usabilidad.htm

Pérez, P. G. *Principios básicos del desarrollo seguro.* División de Seguridad Lógica de Germinus.

Pressman, R. S. (2001). *Ingeniería del Software. Un enfoque práctico. Quinta Edicion.*

Sommerville, I. (2005). *Ingeniería del software. Séptima Edición.* Madrid: Pearson Educación S.A.

UCI. (noviembre de 2008). *La producción en la UCI*. Recuperado el enero de 2009, de Portal UCI - Universidad de la Universidad de las Ciencias Informáticas: http://www.uci.cu/?q=node/46

Anexo 1. No conformidades detectadas en las pruebas a la documentación.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equipo Desarrollo
Diccionario de Datos.	1	El nombre de las variables, o todas en español o en ingles.	Documento Diccionario de datos, Epígrafe juegos de Datos, Adicionar Problema.	Pruebas a la documentación.	S	RA(26-01)	
Manual de Usuario.	2	Falta de concordancia en la redacción.	Documento Manual de Usuario, pág. 6 párrafo 4, primera oración.	Pruebas a la documentación.	S	RA(26-01)	
Manual de Usuario.	3	La página no presenta el pie de página que tiene el documento.	Documento Manual de Instalación, pág. 4,5,8	Pruebas a la documentación.	S	RA(26-01)	
Manual de Usuario.	4	Falta de concordancia en la redacción.	Documento Manual de Usuario, Pág. 8 Párrafo 2, 1ra oración.	Pruebas a la documentación.	S	RA(26-01)	
Manual del Desarrollador	5	Falta de concordancia en la redacción.	Documento Manual del Desarrollador Pág. 3, último párrafo, 1ra oración.	Pruebas a la documentación.	S	RA(26-01)	
Manual del Desarrollador	6	Falta de ortografía a la hora de dividir la palabra: “actualización”	Documento Manual del Desarrollador Pág. 9, 2do párrafo, 2da oración.	Pruebas a la documentación.	NS	RA(26-01)	
Manual del Desarrollador	7	Falta de ortografía a la hora de escribir la palabra: “continuación”	Documento Manual del Desarrollador Pág. 11, último párrafo, última oración.	Pruebas a la documentación.	S	RA(26-01)	
Manual del Desarrollador	8	Falta de concordancia en la redacción.	Documento Manual del Desarrollador Pág. 12, 3er párrafo, final.	Pruebas a la documentación.	NS	RA(26-01)	

Especificación de Requisitos.	9	Palabra mal escrita (emplementacion).	Documento Manual de Especificación de Requisitos, Capitulo Interfaces, Interfaces Software.	Pruebas a la documentación.	S	RA(26-01)
Especificación de requisitos.	10	Tabla de referencia vacía, si no presenta, informárselo al usuario por escrito.	Documento especificación de requisitos, referencias.	Pruebas a la documentación.	R	PD
Manual de Instalación.	11	Se recomienda cambiar los encabezamientos para la instalación del cliente y el servidor, debe ser más amigable con el usuario.	Documento Manual de Instalación	Pruebas a la documentación.	R	PD
Manual de Usuario	12	Se recomienda hacer las descripciones de los pasos a realizar de una misma manera (estándar). Algunas veces ponen: “Realizar una de las siguientes opciones:” y a través de viñetas enuncian las diferentes opciones; y también aparecen las diferentes opciones a continuación de los “:”.	Pág. 29, Última descripción de pasos: “Para descargar una solución debe realizar los siguientes pasos:”	Pruebas a la documentación.	R	RA(26-01)
Manual de Usuario	13	Se recomienda incluir en los caminos (posibles para hacer llevar a cabo alguna función), las combinaciones de teclas que puedan ser parte de estos caminos.	Donde se hace referencia a los distintos caminos para llevar a cabo una función.	Pruebas a la documentación.	R	RA(26-01)
Manual de Usuario	14	Se recomienda especificar siempre todas las vías posibles para llevar a cabo alguna función, hay lugares donde se ponen todas y en otros no.	Donde se hace referencia a los distintos caminos para llevar a cabo una función.	Pruebas a la documentación.	R	RA(26-01)

Anexo 2. No conformidades detectadas en las Pruebas Funcionales.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	Al ocurrir un error en la autentificación se muestra un mensaje de “información” no de error como se describe en el caso uso (CU).	Ventana de autentificación.	Pruebas Funcionales.	S	RA (28-01)	
Aplicación	2	No guarda la configuración (IP, puerto RMI, puerto Socket) de acceso al sistema, como describe el CU	Ventana de autentificación, al seleccionar guardar la configuración.	Pruebas Funcionales.	S	RA (28-01)	
Aplicación	3	Poner correctamente el nombre del botón conectar.	Ventana de Autentificación en el sistema	Pruebas Funcionales.	S	NP	El botón conectar tiene el nombre completo. Depende de la JVM que se utilice para que el mismo aparezca.
Aplicación	4	Al crear un grupo con datos inconsistente la aplicación no muestra ningún mensaje error.	Menú general, Grupo, Nuevo grupo.	Pruebas Funcionales.	S	NP	Lo que sucede es que el componente utilizado para entrar la prioridad y la cuota no registra la entrada de los datos hasta no presionar “enter” o utilizar el spinner.
Aplicación	5	Al modificar la prioridad de un grupo o la cuota el botón aplicar esta deshabilitado.	Menú general, todos los grupos. (Modificar un grupo de la lista).	Pruebas Funcionales.	S	NP	Lo que sucede es que el componente utilizado para modificar la prioridad y la cuota no registra la entrada de los

							datos hasta no presionar “enter” o utilizar el spinner.
Aplicación	6	No se puede verificar si el grupo existe, para que lance un el mensaje de error.	Menú general, grupos.	Pruebas Funcionales.	S	NP	Siempre se verifica la existencia del grupo en el servidor del sistema, para realizar cualquier operación con el mismo.
Aplicación	7	Al cambiarle el privilegio a un usuario como “Administrador” se le asigna todos los problemas a éste sin presionar los botones “Apply” o “OK”.	Management>General>Users(clic derecho sobre un usuario: “propieties”) .	Pruebas Funcionales.	S	NP	El usuario administrador accede a todos los problemas y no son asignados hasta que no se presione el botón “OK” o “Apply”. Sólo se muestran en la interfaz para que no exista inconsistencias
Aplicación	8	Al crear un nuevo “IP Range” y se introduce el IP inicial mayor que el final, se muestra un mensaje de información “Mal Confeccionado”	Management>General>IP Ranges (al crear nuevo “IP Range”).	Pruebas Funcionales.	S	RA (28-01)	
Aplicación	9	Al eliminar un rango de ip el mensaje de confirmación de eliminación muestra “iprange” con minúscula.	General, IP ranges All IP ranges.	Pruebas Funcionales.	S	RA (28-01)	
Aplicación	10	Al insertar un usuario “ya existente en un grupo” **en otro grupo**, el sistema lo muestra en la lista de miembros, sin verificar “si existe en el sistema”.	Management>General>Groups (al insertar un usuario).	Pruebas Funcionales.	S	NP	Porque el usuario no se adiciona hasta que no se presiona el botón “OK” o “Apply”
Aplicación	11	Mantener deshabilitado el botón “connect” mientras los campos a llenar estén vacios.	Ventana de Autentificación en el sistema	Pruebas Funcionales.	R	PD	Se tendrá en cuenta para la segunda versión de la aplicación.

Aplicación	12	Se recomienda que el mensaje de error que se muestra a la hora de crear un nuevo “IP Range” con un rango ya existente, sugiera dónde está el problema.	Management>General>IP Ranges (al crear nuevo “IP Range”).	Pruebas Funcionales.	R	RA (28-01)	
Aplicación	13	Se recomienda informarle al usuario cuando se active (o esté activa) la tecla de la mayúscula, cuando vaya a autentificarse ó a cambiar la contraseña.	Ventana de Autentificación en el sistema. Ventana para cambiar la contraseña.	Pruebas Funcionales.	R	PD	Se tendrá en cuenta para la segunda versión de la aplicación.
Aplicación	14	Se recomienda utilizar un componente que no sea editable a la hora de modificar la prioridad de un grupo o la cuota.	Menú general, todos los grupos. (Modificar un grupo de la lista).	Pruebas Funcionales.	R	PD	Se tendrá en cuenta para la segunda versión de la aplicación.
Aplicación	15	Se recomienda que al insertar un usuario “ya existente en un grupo” **en otro grupo**, el sistema verifique la existencia del usuario antes de mostrarlo en la lista de miembros, para que el cliente no deba esperar presionar el botón “Ok” para darse cuenta de que ya existe.	Management>General>Groups (al insertar un usuario).	Pruebas Funcionales.	R	PD	Se tendrá en cuenta para la segunda versión de la aplicación.

Anexo 3. Consultas utilizadas en las Pruebas de Carga.

No. Consulta	Código
1	SELECT `groups`.name as 'Grupo', `users`.name as 'Usuario' FROM `users` INNER JOIN `user_problem` ON (`users`.userID = `user_problem`.userID) INNER JOIN `problem` ON (`user_problem`.probID = `problem`.probID) INNER JOIN `groups` ON (`users`.groupID = `groups`.groupID) WHERE (`problem`.probID = 1)
2	SELECT `users`.name, `solution`.name, `problem`.probID FROM `users` INNER JOIN `solution` ON (`users`.userID = `solution`.userID) INNER JOIN `user_problem` ON (`users`.userID = `user_problem`.userID) INNER JOIN `problem` ON (`user_problem`.probID = `problem`.probID) WHERE (`users`.nick = 'root')

Anexo 4. Lista de Chequeo aplicada al sistema.

Seguridad

#	Nivel	Evaluación	Eval.	NP	Comentario
1	!!	¿El sistema cuenta con un proceso de autenticación?	5		
2	!!	¿El Acceso al control de protección está incorporado en el sistema?	5		
3	!	¿Todas las contraseñas del equipo de prueba y desarrolladores fueron borradas?	5		
4	!	¿Toda la privacidad, la libertad de información, sensibilidad y consideraciones de la clasificación fueron identificadas, resueltas y establecidas?	5		
5	!!	¿Todos los usuarios tienen asignados un rol en el sistema?	5		
6	!!	¿Todos los roles tienen definidos niveles de acceso en al sistema?	5		
7	!!	¿Se encripta el trafico de información entre el servidor y el cliente utilizando algún protocolo?	5		
Evaluación Numérica		**35**	**Final**		**5**

Portabilidad

#	Nivel	Evaluación	Eval.	NP	Comentario
1	!	¿Existe independencia del ambiente de hardware? (características particulares del hardware)	5		
2	!	¿Existe independencia del ambiente de software? (sistema operativo, lenguaje de programación)	5		
Evaluación Numérica		10		Final	5

Usabilidad

#	Nivel	Evaluación	Eval.	NP	Comentario
1	!	¿Se consideran otros idiomas para las instrucciones en la pantalla de forma adecuada?	2		La aplicación esta implementada en un solo idioma.
2	!	¿Es simple el vocabulario utilizado en el sistema?	5		
3	!	¿Se proporciona tiempo suficiente para realizar las entradas por teclado?	5		
4	!!	¿El sistema es fácil de manejar para usuarios con poca experiencia en el trabajo con computadoras?	4		
5	!	¿El sistema cuenta con una ayuda?	2		El sistema no cuenta con una ayuda interna.
6	!	¿Resulta fácil instalar el software?	5		
7	!!	¿El sistema tiene un ambiente sencillo?	5		
8	!	¿Se entienden la interfaz y su contenido?	4		
9	!	¿Resulta fácil especificar un objeto o una acción?	5		
10	!	¿Resulta fácil entender el resultado de una acción?	5		
11	!	¿Es fácil de utilizar el sistema en la realización de tareas?	4		
12	!!	¿El sistema cuenta con un Manual de Usuario como ayuda para trabajar con la aplicación?	5		
13	!!	¿El sistema cuenta con un Manual de Desarrollo donde se explica cómo programar, montar aplicaciones y usar las librerías necesarias?	5		
14	!!	¿Actúa el sistema en la información de los errores?	5		
15	¡	¿Es específico el sistema en la información de errores?	5		
16	!	¿Actúa el sistema en la corrección de errores?	3		
17	!	¿Se permite la utilización del ratón o el teclado?	4		
18	!	¿Se utiliza mensajes y textos descriptivos?	5		
19	!!	¿Permite una cómoda navegación dentro del producto y una fácil salida de éste?	5		
20	!	¿Se reconoce todas las tareas del software en	5		

		cualquier momento?			
21	**!!**	¿Se proporciona información visual de dónde está el usuario, qué está haciendo y qué puede hacer a continuación?	**4**		
22	**!**	¿Proporciona funciones deshacer, rehacer?	**2**		El sistema no permite ninguna de estas opciones.
23	**!**	¿Existe atajos de teclado bien hechos?	**5**		
24	**!**	¿Se presenta al usuario la información que sólo necesita?	**5**		
Evaluación Numérica		**97**		**Final**	**4.01**

Confiabilidad

#	Nivel	Evaluación	Eval.	NP	Comentario
1	**!**	¿Se recupera el software ante fallas?	**5**		
2	**!**	¿La recuperación ante fallas se realiza en el tiempo requerido para la aplicación?	**5**		
3	**!!**	¿El sistema garantiza que el problema será resuelto, independientemente de cualquier problema ajeno que exista?	**5**		
4	**!!**	¿El sistema garantiza su funcionamiento durante las 24 horas del día y los 7 días de la semana?	**5**		
Evaluación Numérica		**20**		**Final**	**5**