Universidad de las Ciencias Informáticas
Facultad 1

Título: *Estrategia de pruebas de software para el proyecto Fuerza de Trabajo Calificada*

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

Autores:
Yailen Rondón Calzadilla
Yaelsy Tamayo Echemendia

Tutores:
Ing. Eileén Llano Castro
Ing. Yisel Avila Portales

Ciudad de la Habana

Junio, 2010

"No busques ser alguien de éxito, sino busca ser alguien valioso:
lo demás llegará naturalmente."

Albert Einstein

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 _____ del año _____.

_____ _____

Firma del autor **Firma del autor**

_____ _____

Firma del tutor **Firma del tutor**

Datos de contacto

Yailen Rondón Calzadilla

Correo: yrcalzadilla@estudiantes.uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba.

Yaelsy Tamayo Echemendia

Correo: ytechemendia@estudiantes.uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba.

Ing. Eileén Llano Castro

Ingeniera en Ciencias Informáticas.

Correo: ellano@uci.cu , teléfono 835-8863

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba.

Dedicatoria

A mi mamá,

¡Esta es su obra!

Yailen Rondón Calzadilla

A mi mamá,

Por darme alas para perseguir mí sueño.

Yaelsy Tamayo Echemendia

Agradecimientos

A mi mamá, que ha sido siempre la guía de mis logros y me ha puesto por delante de todo, olvidándose de ella misma.

A mi papá, que silenciosamente ha sabido darme todo su apoyo y comprensión.

A Yani, quien no ha faltado nunca para extenderme la mano y darme aliento.

A mi abuela y mi hermano, que sin saberlo han sido parte de lo que me da ánimos para seguir.

A Lecnier, que me ayudó en mis primeros años de la carrera y a pasar la PNP que tanto me costó.

A mis compañeros del grupo 4, que, aunque nos hayan separado, siempre han estado y están ahí para lo que necesite. ¡Han sido el mejor grupo que cualquiera pudiera tener!

A Tony, que en mi último año ha sido mi gran apoyo en los momentos difíciles.

A la Chiqui, mi querida compañera de tesis y de cuarto, que me ha soportado en mis días malos y buenos.

A nuestra tutora, que casi sufre un infarto con nosotras, pero su empeño nos ayudó a llegar hasta aquí.

A mis compañeras de cuarto, en especial Katia que siempre está ahí para darnos ánimos.

A todos aquellos que de una manera u otra han estado apoyándome y que hoy, ingratamente, no relaciono sus nombres, pero que conste que les agradezco por ser parte de algunos episodios de mi vida que no permitiré que mi mala memoria elimine. A todos: ¡Gracias!

Yailen Rondón Calzadilla

Agradecimientos

A la Revolución Cubana y en especial al comandante, por haberme dado la posibilidad de formarme como lo que soy, y de realizar mis sueños.
A mi madre querida, por su perseverancia y su fe en mí, su apoyo y amor incondicional, por estar siempre a mi lado, y por ser mi inspiración y no rendirse jamás conmigo.

A mi familia, en especial a mi abuelo, por tener siempre listo ese consejo que me ayuda a mejorar como persona y a servirme de guía, a mi tía Tamara por quererme y apoyarme tanto, a mi prima Fe María, a mis hermanos Carlitos y Yoel, mi padrastro Miguel y a mi primo Misael, que también se gradúa este año y siempre hemos tenido caminos similares.

A mi mejor amiga y hermana Raikenia por quererme y confiar en mí y a Gady.

A Yisell, porque ha sido una gran amiga y por haber estado siempre dispuesta cuando lo necesité.

A todos mis amigos recién graduados que me han apoyado todo este tiempo, en especial a Dayron, Adiary, Yaisel, Marlén, Ronaldo, Walber, Sotes, Olga, Maikel y Yohana.

A mis compañeros del grupo 4 que los quiero con el corazón, no sé cómo podré convivir sin ellos, en especial María y Pedro, Elizabeth por ser una excelente amiga y ayudarme tanto para la PNP, Dayana, la ruidosa Yailin, Dariel, Luis Ernesto, Javier y Alain.
A mis profes preferidos que han sabido apoyarme y ayudarme en mis cinco años, la profe Matilde y el profe Joel y a Maye, por poder sobrevivir a mí.

A Yasser por soportar mis malcriadeces todo este tiempo, por creer en mí y darme ánimos. Te quiero. Y a todos los compañeros de su apartamento David, Raidel, Ismel, Javier, Anier y Noel.

A mis compañeras de apartamento, por su terrible esfuerzo por soportarme, sobre todo a Katia por su enorme entusiasmo ante todo.

A mi compañera de tesis, yayi, gracias por tantas cosas vividas, por todo ese tiempo inolvidable y por tu amistad. Te quiero, nunca te voy a olvidar.

A nuestra tutora, que no va a poder olvidarse de nosotras, si no es porque nos extraña, es porque no sé de qué manera sobrevivió a estos meses.

Resulta extremadamente difícil agradecerle a cada una de las personas que de una forma u otra hayan contribuido con este momento tan especial en mi vida, pero si de algo sirve, pido perdón si me falta algún nombre, muchas gracias a todos mis amigos, por soportarme tanto y acompañarme en las buenas y en las malas. A todas las personas que he conocido, pues de una manera u otra han hecho de mí lo que soy.

Gracias a todos, están cerca de mi corazón, Uds. han hecho posible que esta personita se convirtiera finalmente en una ingeniera.

Yaelsy Tamayo Echemendia

Resumen

El presente trabajo de diploma surge con motivo de contribuir al aseguramiento de la calidad del producto Sistema Unificado de Gestión de Fuerza de Trabajo Calificada (GeForza), pues para el desarrollo del mismo en el proyecto Fuerza de Trabajo Calificada (FTC), no se tenía definida una estrategia que garantizara las pruebas al producto durante su ciclo de desarrollo. Esta forma de trabajo provoca que se encuentren gran cantidad de errores en las pruebas de liberación provocando que el proceso resulte bastante engorroso pues se realizarían varias iteraciones para erradicar los defectos encontrados, lo que afecta los tiempos de entrega planificados con el cliente y la calidad del producto.

Para ello, se define una estrategia de pruebas a seguir para organizar y documentar las pruebas aplicadas al producto GeForza y los resultados arrojados por las mismas, haciendo uso de los artefactos que se proponen.

Se planificaron pruebas para cada uno de los cuatro niveles establecidos, las cuales fueron aplicadas a los módulos “Administración” y “Sistema Matriz”, utilizando como apoyo, en varios casos, herramientas que automatizaron el proceso. Se obtuvieron, además, una serie de artefactos como el plan de pruebas, diseños de casos de prueba, registros de pruebas, evaluaciones de pruebas, un listado de las no conformidades detectadas, entre otros. Los resultados de la aplicación de las pruebas, registrados en los diferentes artefactos, permitieron arribar a la conclusión de que el producto GeForza no está listo para ser sometido a pruebas de liberación.

Palabras claves: calidad, pruebas de software, GeForza, estrategia de pruebas.

Índice

Introducción

El desarrollo tecnológico creciente que vive el mundo actual ha provocado que las empresas automaticen sus procesos con el objetivo de mejorar la productividad y la calidad de los servicios brindados a sus clientes. Con este aumento de la producción de software, la competencia se hace cada vez más fuerte, motivando a la toma de medidas que garanticen que los productos tengan la mayor calidad.

Cuba ha optado por insertarse también en el mercado del software teniendo como principal exponente la Universidad de las Ciencias Informáticas (UCI), surgida en septiembre del 2002 como iniciativa de nuestro Comandante en Jefe Fidel Castro. La UCI es una universidad productiva donde se desarrollan productos y soluciones tecnológicas integrales.

Uno de los productos desarrollados en la UCI es el Sistema Unificado de Gestión de Fuerza de Trabajo Calificada (GeForza), el mismo está siendo desarrollado por el proyecto Fuerza de Trabajo Calificada (FTC) en la Facultad 1, con el fin de modernizar los sistemas automatizados con los que se llevan a cabo los planes en el Departamento de Fuerza de Trabajo Calificada del Ministerio de Economía y Planificación. La exitosa terminación del proyecto FTC depende, en gran medida, de la gestión de la calidad que se haga y de una adecuada aplicación del proceso de pruebas, de forma organizada y sistemática al producto GeForza, para garantizar que en el proceso de liberación se encuentre un número reducido de defectos y que el producto cumpla con los requisitos especificados por el cliente.

Actualmente en el proyecto FTC no se tiene establecida una correcta planificación para realizarle las pruebas al producto GeForza, se espera llegar al final para la realización de las mismas, lo que trae consigo que se encuentren errores al culminar el desarrollo que pudieron ser erradicados en iteraciones tempranas. Esta forma de trabajo hace que el proceso de liberación del software se torne engorroso, pues al encontrar en esta etapa gran número de defectos, es necesario ejecutar mayor número de iteraciones, lo que ocasiona retrasos en la liberación del mismo, afectando los tiempos de entregas planificados con el cliente y la calidad del producto.

Teniendo en cuenta lo anteriormente planteado el **problema** a resolver queda formulado por la siguiente interrogante: ¿Cómo mejorar el proceso de pruebas en el proyecto FTC para obtener un producto con mayor calidad?

Para dar solución a la problemática planteada el **objeto de estudio** de la presente investigación es la calidad de software y el **campo de acción** lo constituye el proceso de pruebas en el proyecto FTC.

Para dar solución a la problemática descrita se ha planteado como **objetivo general** de la investigación definir e implementar una estrategia de pruebas para detectar los defectos del producto GeForza.

La **hipótesis** que se plantea es que si se define e implementa una estrategia de pruebas en el proyecto FTC entonces se garantizará un producto con un número reducido de defectos.

Variables:

Variable independiente: estrategia de pruebas de software.

Variable dependiente: número reducido de defectos.

Tabla # 1: Operacionalización de las variables de la hipótesis. Fuente: elaboración propia.

Variables	***Dimensión***	***Indicador***	***Sub-indicadores***	***Unidad de medida***
Estrategia de pruebas de software.	*Proyecto FTC.*	*Satisfacción de las necesidades del proyecto.*	*Muy bajo*	*1*
			Bajo	*2*
			Regular	*3*
			Bien	*4*
			Excelente	*5*
		Posibilidad de aplicación.	*Muy bajo*	*1*
			Bajo	*2*
			Regular	*3*
			Bien	*4*
			Excelente	*5*
		Mejora del proceso de prueba.	*Muy bajo*	*1*
			Bajo	*2*
			Regular	*3*
			Bien	*4*
			Excelente	*5*
Número reducido de defectos.	*Producto GeForza.*	*Idoneidad.*	*Completitud de la implementación funcional cercana a 0.*	*0*
			Completitud de la implementación funcional cercana a 1.	*1*

			Cobertura de la implementación funcional cercana a 0.	*0*
			Cobertura de la implementación funcional cercana a 1.	*1*
		Funcionalidad	*Conformidad con la funcionalidad cercana a 0.*	*0*
			Conformidad con la funcionalidad cercana a 1.	*1*
			Conformidad de las interfaces con las normas cercana a 0.	*0*
			Conformidad de las interfaces con las normas cercana a 1.	*1*
		Madurez	*Intensidad de fallos cercano a 1.*	*0*
			Intensidad de fallos cercano a 0.	*1*
			Erradicación de fallos cercano a 0	*0*
			Erradicación de fallos cercano a 1.	*1*
		Rendimiento	*Tiempo de respuesta muy lento.*	*1*
			Tiempo de respuesta lento.	*2*
			Tiempo de respuesta medio.	*3*
			Tiempo de respuesta rápido.	*4*
			Tiempo de espera	*1*

			muy extenso.	
			Tiempo de espera extenso.	*2*
			Tiempo de espera medio.	*3*
			Tiempo de espera corto.	*4*

Con el fin de cumplir el objetivo se plantean las siguientes **tareas**:

- Realización del marco teórico de la investigación.
- Búsqueda de información sobre los diferentes tipos de pruebas que se pueden realizar y escoger las que se ajusten a las características del producto GeForza.
- Búsqueda de información sobre las diferentes herramientas de automatización de pruebas que existen y escoger las que más se ajusten a las características del producto GeForza.
- Establecimiento de una estrategia de pruebas que se ajuste a las necesidades del producto GeForza.
- Ejecución de las pruebas de forma organizada a los módulos “Administración” y “Sistema Matriz” del producto GeForza.
- Documentación de los resultados de las pruebas realizadas a los módulos “Administración” y “Sistema Matriz” del producto GeForza.
- Evaluación de los resultados de las pruebas realizadas a los módulos “Administración” y “Sistema Matriz” del producto GeForza.

Métodos científicos:

Métodos teóricos:

- Analítico-Sintético: este método permite estudiar la calidad de software y buscar información referente a los tipos de pruebas de software existentes, así como las herramientas y métodos que se utilizan para la aplicación de las mismas.
- Inductivo-Deductivo: este método permite llegar a conclusiones lógicas a partir de los conocimientos adquiridos con anterioridad, y así poder plantear la estrategia de pruebas de software para el proyecto FTC.
- Análisis Histórico-Lógico: este método permite estudiar cómo ha evolucionado el proceso de calidad de software y la aplicación de pruebas en los distintos tipos de proyecto.

Métodos empíricos:

- Entrevista: este método permite obtener información de las pruebas que se realizan en la UCI mediante una entrevista al personal encargado de la realización de pruebas en el grupo de Calidad UCI.
- Observación: este método permite recoger información de la realización de las pruebas en situaciones reales, permitiendo obtención de conocimientos del tema a partir de situaciones dadas.

El contenido de la investigación estará plasmado en tres capítulos:

Capítulo 1 "Fundamentación Teórica" se abordará toda la fundamentación teórica realizada para dar a conocer el tema que se estará tratando, o sea, se brindarán todos los conceptos importantes como calidad, pruebas de calidad, tipos de pruebas de calidad de software, metodologías a utilizarse, herramientas y otros conceptos con los que se trabajarán a lo largo de toda la investigación.

Capitulo 2 "Propuesta de solución" se expondrá la propuesta de solución a la problemática planteada, elaborando un procedimiento de pruebas de calidad para el producto GeForza.

Capítulo 3 "Aplicación de la propuesta" recogerá los resultados que arroje la aplicación de la estrategia propuesta, documentándose el "Plan de pruebas" que guiará el proceso, los diseños de casos de prueba para los distintos niveles y una lista de las no conformidades encontradas en el producto y la situación en que se encuentren las mismas.

Capítulo 1: Fundamentación Teórica

1.1. Introducción

En este capítulo se tratan aspectos generales de la calidad de software, la metodología RUP y lo que plantea sobre la realización de las pruebas. Se explican los diferentes tipos de pruebas que existen, sus características y las herramientas automatizadas para realizarlas, así como información relevante para la formulación de una correcta estrategia de pruebas de software. Además, se expone, de forma resumida, las características del producto GeForza desarrollado por el proyecto FTC.

1.2. Calidad de software

La calidad es un factor cada vez más importante en toda tarea o trabajo, y tiene múltiples significados. La calidad de un producto o servicio, casi siempre, se mide por el grado de conformidad y aceptación del cliente que lo recibe. Por su parte, la calidad de software, es un factor que ha robado la atención de muchos autores, obteniéndose así, varios conceptos de calidad de software, por ejemplo:

- "La concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente". (Pressman, 2005)
- "Debe ser caracterizada por su portabilidad, confiabilidad, usabilidad, funcionalidad y mantenibilidad". (NC-ISO-IEC 209126, 2005)
- "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)
- "Conjunto de propiedades y características de un producto o servicio que le confieren su aptitud para satisfacer necesidades explícitas o implícitas". (ISO 8402,1986)
- **"**Grado en el que un conjunto de características inherentes cumple con los requisitos". (ISO 9000, 2000)

A partir de los conceptos citados anteriormente se puede observar que la calidad está directamente vinculada con la satisfacción de los requerimientos del software y se alcanza cuando el servicio o producto satisface por completo las demandas del cliente. Por lo que se llega a la conclusión que la calidad de software es el grado de cumplimiento con los requisitos funcionales y no funcionales del software, establecidos por el cliente y es medible según el grado de satisfacción que muestre este con el producto obtenido.

1.3. Metodologías de desarrollo de software

El éxito del proceso de desarrollo de software depende de la elección correcta de la metodología de desarrollo, la cual se puede definir como "un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software". (Fernández-Medina, 2007)

Según la filosofía de desarrollo, las metodologías de desarrollo de software se clasifican en tradicionales o pesadas y ligeras o ágiles. Los métodos ágiles o ligeros ponen vital importancia en la capacidad de respuesta a los cambios y a mantener una buena relación con el cliente para alcanzar su objetivo, dentro de estas metodologías podemos encontrar a Programación Extrema (XP), *Wicked Problems Righteous Solutions* (SCRUM), Método de Desarrollo de Sistemas Dinámicos (DSDM), Desarrollo de Software Adaptativo (ASD), *Feature Driven Development* (FDD), y *Lean Development* (LD). Por su parte, los métodos tradicionales o pesados son aquellos que alcanzan su objetivo llevando todo el proceso por medio de orden y documentación. Ejemplos de estos son el Proceso Unificado de Desarrollo (RUP) y Marco de Trabajo para Software de Microsoft (MSF).

A continuación se hace un análisis del proceso de pruebas que se lleva a cabo según RUP, dado que esta fue la metodología seleccionada por el analista principal del proyecto FTC, al cual va dirigida la propuesta de estrategia de pruebas de software de esta investigación.

1.4. Pruebas de software según RUP

"Las pruebas de software son un factor importante para asegurarse de que se va por el camino correcto para alcanzar la calidad esperada. Son los procesos que permiten verificar y revelar la calidad de un producto. Estas se integran dentro de las diferentes fases del ciclo de vida del software. Las pruebas de software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación". (Pressman, 2005)

Glen Myers en "*The Art of Software Testing*" (Myers, 1979) establece varias normas sobre la realización de las pruebas:

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Además, esta etapa implica:

- Verificar el funcionamiento de los componentes.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

La prueba es un proceso que se enfoca en el comprobar el correcto funcionamiento del software y el cumplimiento de este con los requisitos establecidos. De ahí que una prueba sea exitosa cuando encuentra un error que no había sido detectado anteriormente.

Para garantizar la calidad del producto, las pruebas de software se deben planificar y ejecutar durante todo el ciclo de vida del software; es por ello que existen diferentes tipos de pruebas de software para las distintas etapas del desarrollo de software.

El proceso unificado es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. (Jacobson, 2000)

Según RUP la prueba es una disciplina en el proceso de ingeniería de software cuyo objetivo es integrar y poner a prueba el sistema. Para desarrollar las pruebas de software define dos tipos de datos:

- Datos reales: permiten probar ocurrencias y casos reales que se presentan, pero que no ocurren en el sistema en desarrollo o en el período de prueba de los sistemas en funcionamiento.
- Datos artificiales: son los que se crean artificialmente tratando de considerar todas las combinaciones y rutas posibles, deben ser preparados por personas distintas de las que programaron el sistema.

RUP define cuatro niveles de prueba: unidad, integración, sistema y aceptación, dentro de los cuales se encuentran diferentes tipos de pruebas, como son:

- Pruebas de unidad: se realizan durante la fase de construcción, específicamente en el flujo de trabajo de implementación; las cuales se basan en probar los componentes implementados como unidades individuales. Las pruebas de unidad están divididas en dos grupos, pruebas de caja blanca y pruebas de caja negra.

- Pruebas de integración: se llevan a cabo durante la fase de construcción, las mismas involucran a un número creciente de módulos y terminan probando el sistema como conjunto. Estas pruebas se pueden plantear desde un punto de vista estructural o funcional. Verifican que los componentes interaccionan entre sí de un modo apropiado después de haber sido integrados en el sistema. Se toman como casos de prueba los casos de uso del diseño. Para ello se utiliza el diagrama de secuencia correspondiente y se diseñan combinaciones de entrada y salida del sistema que lleven a distintas utilizaciones de las clases y en consecuencia de los componentes, que participan en el diagrama.
- Pruebas de sistema: prueban que el sistema funciona globalmente de forma correcta. Cada prueba del sistema prueba combinaciones de casos de uso bajo condiciones diferentes. Se prueba el sistema como un todo probando casos de uso unos detrás de otros y si es posible, en paralelo. En este nivel existen una gran variedad de pruebas que se utilizan según el interés que se tenga respecto al funcionamiento del software.
- Pruebas de aceptación: se realizan para permitir que el cliente valide y verifique todos los requisitos pactados. Estas pruebas las realiza el usuario final en lugar del responsable del desarrollo del sistema. El cliente es quien impone los requisitos pues quien mejor que él para dar fe de su satisfacción.

Se pueden especificar además otros tipos de prueba para probar el sistema como un todo. Por ejemplo:

- Pruebas negativas: Intentan provocar que el sistema falle para poder así revelar sus debilidades. Se puede inferir que las pruebas de rendimiento y penetración o ataque, se pueden ubicar dentro de la misma.

1.4.1. Pruebas unitarias o de unidad

Las pruebas unitarias aíslan cada parte del programa y muestran que las partes individuales son correctas. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. La idea es escribir casos de prueba para cada función no trivial o método en el módulo de forma que cada caso sea independiente del resto.

Estas pruebas aisladas proporcionan cinco ventajas básicas:

- Fomentan el cambio: las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar refactorización), puesto que

permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.

- Simplifica la integración: puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.
- Separación de la interfaz y la implementación: dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro, a veces usando objetos simuladores (*mock object*) para simular el comportamiento de objetos complejos.
- Los errores están más acotados y son más fáciles de localizar: dado que se prueban pequeñas porciones del software.

Las pruebas unitarias no descubrirán todos los errores del código. Por definición, sólo prueban las unidades por sí solas. Por lo tanto, no descubrirán errores de integración, problemas de rendimiento y otros problemas que afectan a todo el sistema en su conjunto. Además, puede no ser trivial anticipar todos los casos especiales de entradas que puede recibir en realidad la unidad de programa bajo estudio. Las pruebas unitarias sólo son efectivas si se usan en conjunto con otras pruebas de software.

Para la realización de las pruebas unitarias las más utilizadas son las pruebas de Caja Blanca y Caja Negra.

Pruebas de Caja Blanca

Las pruebas de caja blanca se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas.

En los sistemas orientados a objetos, las pruebas de caja blanca pueden aplicarse a los métodos de la clase, pero según varias opiniones, ese esfuerzo debería dedicarse a otro tipo de pruebas más especializadas.

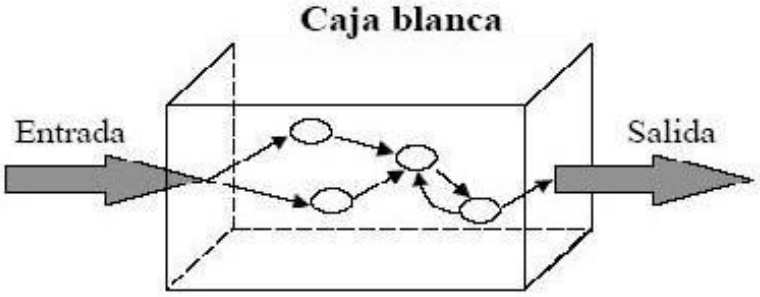

Fig. 1: Enfoque de diseño de pruebas de caja blanca. Fuente: Ingeniería del software, Ian Sommerville.

Capítulo 1: Fundamentación Teórica

De acuerdo con Pressman (Pressman, 2005), la prueba de caja blanca, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba.

Mediante este método el diseñador de pruebas puede obtener casos de prueba que:

- Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales. Ejerciten las estructuras internas de datos para asegurar su validez.

Para la realización de estas pruebas existen diferentes métodos que se analizarán a continuación:

- Prueba del camino básico**:** es una técnica que le permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizarán que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Algunos elementos y conceptos utilizados alrededor de este método son los siguientes:

- Grafo de flujo o grafo del programa: representa el flujo de control lógico de un programa y se utiliza para trazar más fácilmente los caminos de éste. (Cada nodo representa una o más sentencias procedimentales y cada arista representa el flujo de control)
- Complejidad ciclomática: es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto de las pruebas, el cálculo de la complejidad ciclomática representa el número de caminos independientes del conjunto básico de un programa. Esta medida ofrece al probador un límite superior para el número de pruebas que debe realizar para garantizar que se ejecutan por lo menos una vez cada sentencia.
- Camino independiente: cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición.

De forma general, los pasos que se debe seguir para la obtención de los casos de prueba en este método, son los siguientes:

- Emplear el diseño o el código para elaborar el grafo de flujo.
- Determinar la complejidad ciclomática del grafo de flujo.

- Determinar un conjunto básico de caminos linealmente independientes.
- Preparar los casos de prueba que forzarán la ejecución de cada camino del conjunto básico.

➢ Prueba de condición: es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa. Algunos conceptos empleados alrededor de esta prueba son los siguientes:

- Condición simple: Es una variable lógica o una expresión relacional (E_1 < *operador - relacional* > E_2).
- Condición compuesta: Está formada por dos o más condiciones simples, operadores lógicos y paréntesis.

En general los tipos de errores que se buscan en una prueba de condición, son los siguientes:

- Error en operador lógico (existencia de operadores lógicos incorrectos, desaparecidos, sobrantes).
- Error en variable lógica.
- Error en paréntesis lógico.
- Error en operador relacional.
- Error en expresión aritmética.

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

➢ Prueba de bucles: es una técnica que se centra exclusivamente en la validez de las construcciones de bucles (bucles simples, anidados, concatenados y no estructurados).

- Bucles simples. Se les aplica el siguiente conjunto de pruebas:
  - Pasar por alto totalmente el bucle.
  - Pasar una sola vez por el bucle.
  - Pasar dos veces por el bucle.
  - Hacer m pasos por el bucle con $m < n$ (donde n es el número máximo de pasos permitidos por el bucle).
  - Hacer n - 1, n y n + 1 pasos por el bucle.
- Bucles anidados. Si se empleara el mismo enfoque de prueba de bucles simples a los bucles anidados, el número de pruebas aumentaría considerablemente por lo cual se sugiere emplear el siguiente enfoque:
  - Comenzar por el bucle más interior. Establecer o configurar los demás bucles con sus valores mínimos.

  - Llevar a cabo las pruebas de bucles simples para el bucle más interior, mientras se mantienen los parámetros de iteración de los bucles externos en sus valores mínimos. Añadir otras pruebas para valores fuera de rango o excluidos.
  - Progresar hacia fuera, llevando a cabo pruebas para el siguiente bucle, pero manteniendo todos los bucles externos en sus valores mínimos y los demás bucles anidados en sus valores típicos.
  - Continuar hasta que se hayan probado todos los bucles.
- Bucles concatenados. Estos bucles se pueden probar utilizando el enfoque de bucles simples, siempre y cuando cada uno de los bucles sea independiente del resto de lo contrario se debe emplear el enfoque de bucles anidados.
- Bucles no estructurados. Siempre que sea posible estos bucles deben rediseñarse.

Pruebas de Caja Negra

Las pruebas de Caja negra se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene. (Jacobson, Booch, Rumbaugh, 2000)

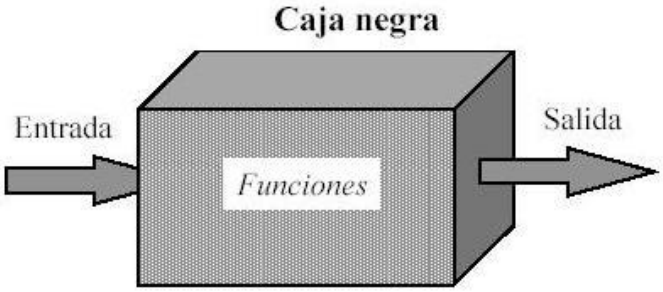

Fig. 2: Enfoque de diseño de pruebas de caja negra. Fuente: Ingeniería del software, Ian Sommerville.

Para Pressman, las pruebas de caja negra se centran en los requisitos funcionales del software. Es decir, permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Se trata de un enfoque que intenta descubrir diferentes tipos de errores que no se encuentran con los métodos de caja blanca.

La prueba de caja negra para Pressman (Pressman, 2005), intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.

- Errores de rendimiento.
- Errores de inicialización y de terminación.

Algunos de los métodos empleados en estas pruebas son los siguientes:

➢ Métodos de prueba basados en grafos: en este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. En este método:

- Se crea un grafo de objetos importantes y sus relaciones.
- Se diseña una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores.

➢ Partición equivalente: Pressman presenta la partición equivalente como un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

El objetivo de partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases.

La partición equivalente es subjetiva. Dos probadores quienes prueban un programa complejo pueden llegar a diferentes conjuntos de particiones.

En el diseño de casos de prueba para partición equivalente se procede en dos pasos:

1. Se identifican las clases de equivalencia. Las clases de equivalencia son identificadas tomando cada condición de entrada (generalmente una oración o una frase en la especificación) y repartiéndola en dos o más grupos.

Es de notar que dos tipos de clases de equivalencia están identificados: las clases de equivalencia válidas representan entradas válidas al programa, y las clases de equivalencia inválidas que representan el resto de los estados posibles de la condición (es decir, valores erróneos de la entrada).

2. Se define los casos de prueba. El segundo paso es el uso de las clases de equivalencia para identificar los casos de prueba. El proceso es como sigue: se asigna un número único a cada clase de equivalencia. Hasta que todas las clases de equivalencia válidas han sido cubiertas por los casos de prueba, se escribe un nuevo caso de prueba que cubra la clase de equivalencia válida. Y por último hasta que los casos de prueba hayan cubierto todas las clases de equivalencia inválidas, se escribe un caso de la prueba que cubra una, y solamente una, de las clases de equivalencia inválidas descubiertas.

- Análisis de valores límite: los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites (AVL) como técnica de prueba. El análisis de valores límite lleva a una elección de casos de prueba que ejerciten los valores límite.

El análisis de valores límite es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida.

Condiciones sublímite: Las condiciones límite normales son las más obvias de descubrir. Estas son definidas en la especificación o son evidentes al momento de utilizar el software. Algunos límites, sin embargo, son internos al software, no son necesariamente aparentes al usuario final pero aún así deben ser probadas por el probador. Estas son conocidas como condiciones sublímites o condiciones límite internas. Una condición sublímite común es la tabla de caracteres ASCII, por ejemplo, si se está evaluando una caja de texto que acepta solamente los caracteres AZ y az, se debe incluir los valores en la partición inválida justo «debajo de» y «encima de» esos caracteres de la tabla ASCII, [", y {.

- Prueba de la tabla ortogonal**:** hay aplicaciones donde el número de parámetros de entrada es pequeño y los valores de cada uno de los parámetros está claramente delimitado. Cuando estos números son muy pequeños (por ejemplo, 3 parámetros de entrada tomando 3 valores diferentes), es posible considerar cada permutación de entrada y comprobar exhaustivamente el proceso del dominio de entrada. En cualquier caso, cuando el número de valores de entrada crece y el número de valores diferentes para cada elemento de dato se incrementa, la prueba exhaustiva se hace impracticable.

La prueba de la tabla ortogonal puede aplicarse a problemas en que el dominio de entrada es relativamente pequeño pero demasiado grande para posibilitar pruebas exhaustivas. El método de prueba de la tabla ortogonal es particularmente útil al encontrar errores asociados con fallos localizados, una categoría de error asociada con defectos de la lógica dentro de un componente software.

La prueba de tabla ortogonal permite proporcionar una buena cobertura de pruebas con bastantes menos casos de prueba que en la estrategia exhaustiva.

- Adivinando el error: dado un programa particular, se obtiene, por la intuición y la experiencia, ciertos tipos probables de errores y entonces se escriben casos de prueba para exponer esos errores. Es difícil dar un procedimiento para esta técnica puesto que es en gran parte un proceso intuitivo.

La idea básica es enumerar una lista de errores posibles o de situaciones propensas a errores y después escribir los casos de prueba basados en la lista. Por ejemplo, la presencia del valor 0 en la entrada de un programa es una situación con tendencia a error. Por lo tanto, puede ser que se escriban los casos de prueba para los cuales los valores particulares de la entrada tienen valor 0 y los valores particulares para los que la salida se colocan de manera forzada a 0.

Basado en el programa de la Universidad de las Ciencias Informáticas (UCI) y lo planteado por RUP el método que se utilizará para la propuesta de la estrategia será el de particiones equivalentes de caja negra, pues es el más conocido y utilizado, además, con este método se comprueba el funcionamiento de cada caso de uso detallando cada conjunto de datos válidos o inválidos que puedan introducirse.

1.4.2. Pruebas de Integración

Las Pruebas de Integración según RUP verifican que los componentes interaccionan entre sí de un modo apropiado después de haber sido integrados en el sistema. Para ello se diseñan combinaciones de entrada y salida del sistema que lleven a distintas utilizaciones de las clases y en consecuencia de los componentes que participan en el diagrama. Consiste en realizar pruebas para verificar que un gran conjunto de partes de software funcionan juntos. Estas pruebas se pueden plantear desde un punto de vista estructural o funcional.

Las pruebas estructurales de integración son similares a las pruebas de caja blanca pero trabajan a un nivel conceptual superior. No se refieren a sentencias del lenguaje sino a llamadas entre módulos. Se trata de identificar todos los posibles esquemas de llamadas y ejercitarlos para lograr una buena cobertura de segmentos o de ramas.

Las pruebas funcionales de integración son similares a las pruebas de caja negra. Con estas se trata de encontrar fallos en la respuesta de un módulo cuando su operación depende de los servicios prestados por otro(s) módulo(s). Las pruebas funcionales se hacen mediante el diseño de casos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático. Según se van acercando al sistema total, estas pruebas se basan cada vez más en la especificación de requisitos del usuario. En todas estas pruebas funcionales se siguen utilizando las técnicas de partición en clases de equivalencia y análisis de casos límite (fronteras).

Existen diferentes tipos de pruebas de integración, los cuales son:

Pruebas funcionales ascendentes:

En este caso se empiezan a probar los componentes más simples, o sea, los más internos (E, F). Luego se prueban los componentes que llaman a estos que ya están probados y se prueban (B, C, D). Por último se combinan con el componente más externo que sería el que los llamaría a todos (A). O sea, se prueban los componentes desde el más interno hasta el más externo, comprobando la respuesta del sistema ante llamadas que involucran a más de un componente.

Ventaja: este tipo de enfoque permite un desarrollo más en paralelo que el enfoque de arriba - abajo, pero presenta mayores dificultades a la hora de planificar y de gestionar.

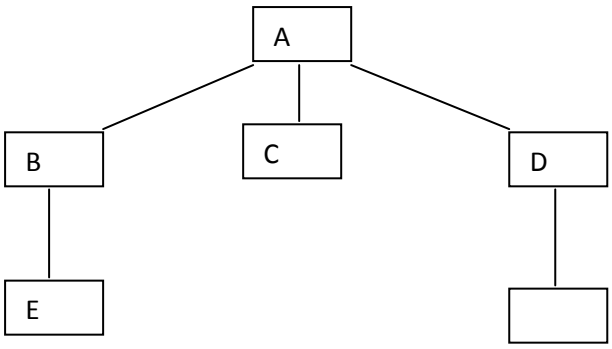

Fig. 3: Enfoque de Pruebas funcionales ascendentes. Fuente: elaboración propia.

Pruebas funcionales descendentes.

Al contrario de las pruebas ascendentes, en este caso se prueba el componente que los integra todos y se simulan los otros de más bajo nivel.

Ventaja: una de las ventajas de aplicar esta estrategia es que las interfaces entre los distintos componentes se prueban en una fase temprana y con frecuencia.

Estrategias combinadas.

A menudo es útil aplicar las estrategias anteriores conjuntamente. De este modo, se desarrollan partes del sistema con un enfoque descendente mientras que los componentes más críticos en el nivel más bajo se desarrolla siguiendo un enfoque ascendente. En este caso es necesaria una planificación cuidadosa y coordinada de modo que los componentes individuales se encuentren en el centro.

Pruebas de humo

De acuerdo con Roger S. Pressman (Pressman, 2005), la prueba de humo es un método de prueba de integración que es comúnmente utilizada cuando se ha desarrollado un producto de software empaquetado. Es diseñado como un mecanismo para productos críticos por tiempo, permitiendo que el equipo de software valore su proyecto sobre una base sólida.

Algunas de las actividades de la prueba de humo son las siguientes:

- Los componentes que han sido traducidos a código se integran en una construcción.
- Se diseña una serie de pruebas para descubrir errores que impiden a la construcción realizar su función adecuada.

En la prueba de humo es habitual que la construcción se integre con otras construcciones y que se aplique una nueva prueba de humo al producto completo. La integración puede hacerse bien de forma descendente o ascendente.

Las pruebas de integración es la fase las pruebas de software en la cual módulos individuales de software son combinados y probados como un grupo. Son las pruebas posteriores a las pruebas unitarias y preceden a las pruebas de sistema.

En el caso de la estrategia propuesta se desarrollarán pruebas funcionales de manera ascendente para probar el funcionamiento de varios componentes integrados, empezando a probar desde lo más simple a lo más complejo.

1.4.3. Pruebas de sistema

Las pruebas de sistema propuestas por la Metodología RUP, prueban que el sistema funciona globalmente de forma correcta. Cada prueba del sistema prueba combinaciones de casos de uso bajo condiciones diferentes. Se prueba el sistema como un todo probando casos de uso unos detrás de otros y si es posible, en paralelo.

Prueba de carga máxima:

Consiste en probar si el sistema puede manejar el volumen de actividades que ocurren cuando el sistema está en el punto más alto de su demanda de procesamiento, tanto en número de transacciones, número de terminales y magnitud de los valores.

Prueba de almacenamiento o de tensión:

Determinar si el sistema puede almacenar una alta cantidad proyectada de datos tanto en sus dispositivos de discos fijos y removibles, y según el diseño y configuración de los archivos.

Prueba de volumen:

Analiza el comportamiento de la aplicación y la base de datos con volúmenes de datos almacenados similares a los esperados en la explotación real del sistema.

Pruebas de recuperación o de disponibilidad de datos:

Probar la capacidad del sistema para recuperar datos y restablecerse después de una falla. Para efectuar estas pruebas se deben previamente sacar copias de respaldo de los archivos reales.

Prueba de usabilidad:

Se determina cómo utilizan los usuarios el sistema al procesar datos o preparar informes. Respecto al sistema: debe ser amigable y de fácil operación, con guía interactiva de orientación al usuario y mensajes indicativos de ocurrencias del sistema. Evaluar, respecto a los usuarios: el nivel de capacitación, el grado de utilización del sistema, su correcta aplicación o uso. Evaluar, respecto al personal de informática: el nivel tecnológico para operar el sistema y la disponibilidad de técnicos con capacidad de mantener el sistema.

Pruebas de instalación:

Algunos tipos de sistemas de software tienen complicados procedimientos de instalación. Las pruebas de los procedimientos de instalación es una parte importante del proceso de prueba del sistema. Esto es particular de un sistema automatizado de instalación que sea parte del paquete del programa. Al funcionar incorrectamente el programa de instalación podría evitar que el usuario tenga una experiencia acertada con el sistema. La primera experiencia de un usuario es cuando él o ella instalan la aplicación. Si esta fase se realiza mal, entonces el usuario/cliente puede buscar otro producto o tener poca confianza en la validez de la aplicación.

Pruebas de configuración:

Programas tales como sistemas operativos, sistemas de gerencia de base de datos, y programas de conmutación de mensajes soportan una variedad de configuraciones de hardware, incluyendo varios tipos y números de dispositivos de entrada-salida y líneas de comunicaciones, o diversos tamaños de memoria. A menudo el número de configuraciones posibles es demasiado grande para probar cada uno, pero en lo posible, se debe probar el programa con cada tipo de dispositivo de hardware y con la configuración mínima y máxima. Si el programa por sí mismo se puede configurar para omitir componentes, o si puede funcionar en diversas computadoras, cada configuración posible de este debe ser probada.

Pruebas de almacenamiento:

Los programas tienen de vez en cuando objetivos de almacenamiento que indiquen, por ejemplo, la cantidad de memoria principal y secundaria que el programa usa y el tamaño de los archivos temporales. Se diseñan casos de prueba para demostrar que estos objetivos de almacenaje no han sido encontrados.

Pruebas de seguridad:

Consisten en verificar los mecanismos de control de acceso al sistema para evitar alteraciones indebidas en los datos.

Pruebas de entorno:

Consisten en verificar las interacciones del sistema con otros sistemas dentro del mismo entorno.

Pruebas de rendimiento:

Consisten en determinar que los tiempos de respuesta están dentro de los intervalos establecidos en las especificaciones del sistema.

Pruebas de prestaciones o de comunicaciones:

Determinan que las interfaces entre los componentes del sistema funcionan adecuadamente, tanto a través de dispositivos remotos, como locales. Asimismo, se han de probar las interfaces hombre/máquina.

Pruebas de *stress*:

Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponibles).

Con el interés de probar que el sistema cumpla con los requisitos y necesidades básicas, se le aplicarán pruebas de carga, de seguridad, de usabilidad, volumen y de *stress*. El resto de las pruebas de sistema, aunque cada una tiene un alto nivel de importancia, comprueban aspectos que para el producto GeForza no son realmente relevantes.

1.4.4. Pruebas de regresión

Las pruebas de regresión son cualquier tipo de pruebas de software que intentan descubrir las causas de nuevos errores (*bugs*), carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software, inducidos por cambios recientemente realizados en partes

de la aplicación que anterior al cambio no eran propensas a este tipo de error. Esto implica que el error tratado se reproduce como consecuencia inesperada del citado cambio en el programa.

Las regresiones son clasificadas en 3 tipos:

- Local: los cambios introducen nuevos errores.
- Desenmascarada: los cambios revelan errores previos.
- Remota: los cambios vinculan algunas partes del programa (módulo) e introducen errores en ella.

En la mayoría de los procesos de desarrollo de software es recomendable que se realice una prueba de regresión luego de introducido un cambio en el programa, ya que este puede haber acarreado nuevos errores. Las pruebas de regresión pueden usarse no solo para probar la corrección de un programa, sino, a menudo, para rastrear la calidad de su salida.

Para la propuesta se solicita realizar las distintas pruebas por iteraciones, que recogerán las deficiencias detectadas en la actual y en la anterior iteración, permitiendo probar no sólo el cumplimiento de los errores pendientes, sino también los cambios realizados. Con esta manera de proceder se evita la necesidad de aplicar pruebas de regresión.

1.4.5. Pruebas de aceptación

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

Las pruebas de aceptación son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al usuario.

La validación del sistema se consigue mediante la realización de pruebas de caja negra que demuestran la conformidad con los requisitos y que se recogen en el plan de pruebas, el cual define las verificaciones a realizar y los casos de prueba asociados.

1.4.6. Roles y artefactos para pruebas de RUP

La metodología RUP propone diferentes roles que tienen como propósito el desarrollo de las pruebas de software. Estos son:

- Administrador de pruebas: es el encargado del éxito y la planificación de las pruebas.
- Analista de pruebas: es responsable de identificar y definir las pruebas, así como supervisar el progreso y los resultados de las mismas.

- Diseñador de pruebas: es responsable de identificar las técnicas y herramientas necesarias, así como definir y estructurar los elementos de prueba y de asegurar la implementación exitosa de estas.
- Probador: es el encargado de ejecutar las pruebas, analizando los resultados obtenidos.

El Proceso Unificado de Desarrollo de Software plantea 7 artefactos fundamentales para el flujo de trabajo de prueba, de estos se obtendrán 4 en la estrategia propuesta, los cuales son:

- Caso de prueba: los casos de prueba son un conjunto de condiciones o variables bajo las cuales el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio.
- Plan de prueba: documento que registra la planificación de los tipos de pruebas, la fecha y la etapa en que se van a realizar.
- Defecto de prueba o no conformidad: documento que plasma los errores encontrados en las pruebas.
- Evaluación de la prueba: verificación de los resultados obtenidos en las pruebas y los esperados según lo planificado.

Para el registro de estos artefactos se utilizará la versión 2.0 del expediente de proyecto de la UCI, vigente en el proyecto FTC desde febrero del 2009, y por tanto las plantillas serán las establecidas en el mismo.

1.5. Herramientas automatizadas para pruebas de software

Existen un sinnúmero de herramientas para automatizar las pruebas de software. A continuación se expondrán algunos ejemplos de las más conocidas dentro del objeto de estudio de la investigación y que por sus características, pueden ser candidatas a ser propuestas para la estrategia.

Específicamente para las pruebas unitarias y de integración existen herramientas como:

Zend Framework:

Se trata de un marco de trabajo para el desarrollo de aplicaciones Web y servicios Web con PHP, que brinda soluciones para construir sitios web modernos, robustos y seguros. Además es código abierto y trabaja con PHP 5. Trabaja con MVC (Modelo Vista Controlador). El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos. Posee completa documentación y pruebas de alta calidad.

Selenium IDE:

Es un *plugin* para Firefox que facilita la labor de realizar juegos de pruebas en aplicaciones web. Para ello permite grabar, editar y correr pruebas. Con la grabación, se puede grabar la navegación realizada en Firefox y después reproducirla. Aunque no es tan solo una herramienta de grabación, además permite editar las grabaciones.

Las características que ofrece son:

- Grabación y reproducción de casos de prueba.
- Selección inteligente de campos mediante sus identificadores, nombres o contraseñas.
- Autocompletado para todos los comandos de Selenium.
- Paso a paso.
- *Debug* y puntos de parada.
- Guardar las pruebas como HTML, etc.

Test Maker:

Es un producto software profesional para organizar pruebas y analizar los resultados. Se puede usar en cualquier esfera en la que se requiera comprobación rápida y eficaz.

Características principales:

- Ayuda a crear exámenes en segundos.
- Crea un número ilimitado de bancos de prueba.
- Crea más de 10 variables de respuesta en cada pregunta.
- Guarda las pruebas creadas en una base de datos.
- Suministra protección por contraseña.
- Tiene la capacidad de especificar límites de tiempo.
- Administración y calificación de pruebas.
- Las pruebas pueden usarse en una sola máquina o en red.

Existen, además, otras herramientas para la realización de pruebas de sistema, entre ellas podemos encontrar las siguientes:

JMeter

El JMeter es una herramienta libre, además es una herramienta Java, que permite realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web. Es una herramienta de carga para llevar acabo simulaciones sobre cualquier recurso de software. De las herramientas gratis, es la más completa y útil para realizar pruebas de carga.

El JMeter muestra los resultados de las pruebas en una amplia variedad de informes y gráficas.

Además facilita una rápida detección de los cuellos de botella existentes debido al tiempo de respuesta excesivo. En una aplicación es de vital importancia emplear algo de tiempo a preparar pruebas de eficiencia bajo carga y stress, antes de ser entregada. El tiempo invertido es recuperado con creces, ya que se detectarán los posibles efectos colaterales y se podrá comprobar si esa nueva funcionalidad soporta la cantidad de usuarios concurrentes que se especificaban en los requisitos.

Es una herramienta que sirve para realizar pruebas funcionales, pero también sirve para realizar pruebas de regresión en aplicaciones web. Tiene una estructura en árbol que le da potencia, permitiendo que sea la imaginación de quien la use la que ponga los límites a la hora de diseñar el plan de prueba. Y brinda mayor cantidad de variantes para recoger los resultados obtenidos, que el resto de las herramientas gratis, lo que permiten hacer un análisis exhaustivo de las pruebas realizadas.

Nessus:

Es el líder mundial en escáneres activos, con alta velocidad de descubrimiento, la auditoría de configuración, el perfil activo, el descubrimiento de los datos sensibles y análisis de la vulnerabilidad de su seguridad. Se pueden distribuir a lo largo de toda una empresa. Nessus es apoyado por un equipo de investigación de renombre mundial y tiene la base de la vulnerabilidad de mayor conocimiento, por lo que es adecuado para los entornos más complejos.

Shadow Security Scanner:

Es un completísimo y efectivo escáner de vulnerabilidades para la plataforma de Windows nativa, aunque también examina servidores de cualquier otra plataforma revelando brechas en *Unix, Linux, FreeBSD, OpenBSD y Net BSD.* Por su arquitectura, Shadow Security Scanner también descubre fallos en CISCO, HP y otros equipos de la red.

Los análisis son muy rápidos y se ofrecen informes de cada una de las vulnerabilidades y sus posibles soluciones, además de actualizarse automáticamente a través de Internet.

Data Generator for PostgreSQL:

EMS Data Generator es una utilidad de gran alcance para generar datos de prueba a varias tablas de base de datos PostgreSQL de una vez. La aplicación asistente permite definir tablas y campos para generar datos, configurar valores de rangos, generar campos *char* por máscara, cargar valores para los campos a partir de archivos, obtener listas de valores desde consultas SQL y muchas otras características para generar datos de prueba de una forma sencilla y directa. También proporciona

aplicación de consola, que le permite generar datos en un solo toque mediante el uso de plantillas de generación.

Entre sus principales características se encuentran:

- Asistente para interfaz de uso fácil.
- Seis idiomas disponibles: inglés, francés, alemán, italiano, ruso y español.
- Generar datos de varias tablas de diferentes bases de datos en un host.
- Todos los datos PostgreSQL admite los tipos, incluyendo *Array,* y los tipos de direcciones de red geométrica.
- Diferentes tipos de generación para cada campo, incluyendo la lista, al azar, la generación de datos adicionales y mucho más.
- Capacidad para utilizar los resultados de consultas SQL como lista de valores para la generación de datos.
- Control automático sobre integridad referencial para las tablas vinculadas de generación de datos.
- Amplia variedad de parámetros de generación para cada tipo de campo.
- Capacidad para establecer los valores NULL para cierto porcentaje de los casos.
- Posibilidad de salvar todos los parámetros de generación, situado en sesión del asistente actual.
- Utilidad de línea de comandos para generar utilizando el archivo de plantilla.

Para la estrategia se propone utilizar el Selenium IDE para automatizar las pruebas funcionales debido a las facilidades que brinda y los conocimientos existentes acerca de sus funcionalidades, además, es, de las herramientas candidatas para ese tipo de pruebas, la más completa.

Del segundo grupo se tomará para la propuesta de la estrategia el JMeter para la realización de pruebas de carga y stress y el Data Generator for PostgreSQL ya que son las más completas de las herramientas para cubrir el nivel de pruebas de sistema y de la cuales existe un mayor conocimiento sobre su uso.

1.6. Producto GeForza

GeForza pretende informatizar los procesos que se llevan a cabo en el departamento Fuerza de Trabajo Calificada del Ministerio de Economía y Planificación. Actualmente, existen tres sistemas que tratan de agilizar esta tarea; en cierto modo lo logran, pero no de manera eficiente, además, los mismos presentan una serie de problemas que los hacen una solución poco eficaz, para ser utilizado en tareas de tan alta importancia. Por lo cual, el propósito que se plantea el proyecto FTC es darle

solución a los problemas anteriormente expuestos a través de un sistema informático, para esto se desarrollan cinco módulos: Sistema Matriz, Sistema Plan de Ingreso, Sistema de Distribución, Administración y Sistema Plan de Ingreso a las SUM.

El producto desarrollado va a ser utilizado por el 75% de las entidades del país, lo que el estado de los ordenadores clientes es casi imposible de obtener de manera concreta, por lo que se prepara el software teniendo en cuenta que será utilizado por ordenadores con bajo rendimiento de hardware. Las computadoras cliente del sistema y ubicadas en el dominio de la organización postal, deben tener un navegador de Internet con base Netscape (Mozilla Firefox).

Las interfaces destinadas al ciudadano, deben de programarse en HTML estándar y ser compatibles a los navegadores *Microsoft Internet Explorer* y *Netscape*. El sistema debe poseer una interfaz amigable al usuario, brindando facilidades que permitan interactuar con el sistema de forma fácil y rápida.

El sistema requiere de un buen rendimiento que se apoya en el mínimo acceso a base de datos, realización de consultas no redundantes, en una transferencia mínima de datos cliente-servidor. Para un funcionamiento óptimo de la aplicación se deben seguir las diferentes técnicas de elaboración en la web, que faciliten el rápido acceso a sus páginas.

La eficiencia del producto debe estar determinada en gran medida por el aprovechamiento de los recursos que se disponen en el modelo Cliente/Servidor, y la velocidad de las consultas en la base de datos. La herramienta propuesta debe ser rápida y el tiempo de respuesta debe ser el mínimo posible, adecuado a la rapidez con que el cliente requiere la respuesta a su petición.

1.7. Estrategia de pruebas de software

El proceso de ejecución de pruebas debe ser considerado durante todo el ciclo de vida de un proyecto, para así obtener un producto de alta calidad. Su éxito dependerá del seguimiento de una estrategia de pruebas bien planificada.

La estrategia de pruebas de software integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba: la planificación, el diseño de casos de prueba, la ejecución y los resultados, tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software.

Para el logro de una exitosa estrategia de pruebas de software se debe tener en cuenta: especificar los requisitos del producto de manera cuantificable mucho antes de que comiencen las pruebas, establecer los objetivos de las pruebas de manera explícita, comprender qué usuarios van a manejar el software y desarrollar un perfil para cada categoría de ellos, desarrollar un plan de pruebas que haga

hincapié en la “prueba de ciclo rápido”, construir un software robusto diseñado para probarse a sí mismo, usar revisiones técnicas formales efectivas como filtros antes de la prueba, llevan a cabo revisiones técnicas formales para evaluar la estrategia de pruebas y los propios casos de pruebas y desarrollar un enfoque de mejora continua al proceso de pruebas.

Las pruebas que se le realicen al software deben estar planificadas con antelación para poder tener un resultado esperado con el cual validar la corrección o no del funcionamiento de la aplicación. Para esto, se desarrolla un plan de pruebas, como parte de la estrategia de pruebas.

La formulación de la estrategia de pruebas de software debe tener en cuenta los pasos que se van a seguir para la realización de las pruebas, como pueden ser, para el caso de esta investigación, los siguientes:

- Identificar las distintas pruebas y las etapas en la que se aplicarán.
- Propuesta del instrumento de medición.
- El diseño y registro de casos de prueba.
- Aplicación de las pruebas propuestas.
- Documentación y evaluación de los resultados de la implementación de la estrategia propuesta.

1.8. Conclusiones

Durante el desarrollo del capítulo se abordaron los temas de interés para la investigación, obteniéndose como conclusiones, que de los tipos de pruebas estudiados fueron seleccionadas, teniendo en cuenta las características que debe cumplir el producto GeForza, pruebas de caja negra utilizando el método de particiones equivalentes para el nivel de unidad, pruebas funcionales ascendentes haciendo uso también del mismo método de caja negra para el nivel de integración, para el nivel de pruebas de sistema se aplicarán carga, stress, usabilidad, volumen y seguridad y el último nivel serán las pruebas de aceptación luego de la liberación del producto. Se obtuvo, además, que las herramientas que automatizarán el proceso serán el Selenium IDE, el JMeter y el Data Generator for PostgreSQL. Por último, quedaron establecidos los pasos que van a guiar la confección de la estrategia a proponer, sentando así las bases para la confección de la propuesta de solución.

Capítulo 2: Propuesta de solución

2.1. Introducción

En este capítulo se describe la estrategia de pruebas de software que se propone para evaluar la calidad del producto GeForza. Se detallan, los artefactos que se generan con la aplicación de dicha estrategia y los roles que intervienen en la confección de cada uno de ellos.

Luego de exponer los componentes de la propuesta, se describe, paso a paso, la aplicación de cada una de las pruebas seleccionadas, con los roles que la llevan a cabo y los artefactos que se generan. Para cada uno de los niveles de pruebas se establece, además, las herramientas que automatizan y agilizan la aplicación de las mismas.

2.2. Definición de la estrategia

Para la aplicación de la estrategia se definen las etapas en las que se aplicarán las pruebas, para ello se tiene en cuenta que las pruebas, según RUP, se aplican en las fases de Construcción y Transición. En la fase de Construcción se aplicarán pruebas unitarias para comprobar el funcionamiento de cada uno de los módulos, pruebas de integración para verificar el funcionamiento integrado de los distintos componentes y pruebas de sistema para probar la correspondencia con las características establecidas. En la fase de Transición se ejecutarán pruebas de aceptación, donde el cliente evaluará su conformidad con el producto final.

Luego de establecer las etapas y los tipos de pruebas que se realizarán se procederá al diseño de cada una de las pruebas propuestas, con los artefactos que se generarán y los roles encargados de cada paso.

Ya diseñadas las pruebas, se implementan según lo planificado y se realiza el registro de los resultados que se van arrojando, con los cuales se evaluará la aplicación de la estrategia.

2.2.1. Alcance

La estrategia se aplicará a todos los módulos del producto realizado, de manera que se pruebe cada parte del sistema, garantizando la calidad del producto GeForza desde los componentes más internos hasta los más externos. Es importante destacar que para validar la estrategia en el presente trabajo de diploma sólo se aplicará a dos de los módulos del producto GeForza: Sistema Matriz y Administración.

2.3. Descripción de roles

La estrategia contempla, basada en RUP, la existencia de tres roles responsables de la aplicación de las pruebas. Estos serán ocupados según la preparación que se posea de acuerdo con la competitividad requerida, o sea, según los conocimientos mínimos que deba poseer la persona. A continuación se relacionan dichos roles y competencias:

Administrador de pruebas: este rol deberá ser ocupado por un experto en pruebas de software, que posea conocimientos de métricas de pruebas, de la metodología RUP, de ingeniería de software, y de los requerimientos del proyecto, o sea, deberá conocer el negocio. Todas estas nociones estarán encaminadas a la mejor planificación de las pruebas y al éxito de las mismas, además, será el encargado de seleccionar las pruebas que se realizarán y monitorear la aplicación de las mismas debido a que incluirá las responsabilidades del analista de prueba, rol que se hace innecesario por la utilización de la estrategia donde ya se establecen las pruebas y técnicas a utilizarse.

Diseñador de pruebas: este rol deberá tener un vasto conocimiento de las herramientas de pruebas, ya que será el responsable de seleccionar las herramientas con las que se realizarán las pruebas. Conocerá, además, de ingeniería de software, de pruebas de software, de la metodología RUP y del negocio, puesto que es el encargado de diseñar los casos de prueba.

Probador: este rol será el responsable de ejecutar las pruebas tal y como fueron planificadas, para ello debe tener conocimientos básicos de la metodología RUP, de las pruebas de software y, sin lugar a dudas, debe ser un especialista en las herramientas seleccionadas para la automatización de las pruebas.

Todas estas competencias, anteriormente planteadas, están encaminadas a conformar el equipo de pruebas, el mismo que será responsable de desarrollar los artefactos que se obtienen de este flujo de trabajo a medida que se va transitando por cada paso de la estrategia.

2.4. Descripción de los artefactos

Como resultado del proceso, para esta estrategia, se obtendrán varios artefactos, los cuales serán formulados como lo establece la versión 2.0 del expediente de proyecto vigente en la UCI y quedarán archivados en el mismo.

A continuación se describe cada uno de los artefactos.

Plan de pruebas:

Esta plantilla será realizada por el administrador de pruebas y contendrá:

- Nombre del proyecto.
- Nombre del producto.

- Versión actualizada del documento.
- Control de versiones: tabla que se actualizará con cada cambio que se realice en el documento.
- Introducción: dentro de este acápite se describirá el alcance del documento, las definiciones, acrónimos y abreviaturas que se utilizan y las referencias hechas a otros documentos.
- Roles y Responsabilidades: se describirán los roles existentes en el proyecto para la realización de las pruebas y las responsabilidades específicas de cada uno.
- Escenario de pruebas: contendrá el diagrama de despliegue y los recursos del sistema con los que cuenta el proyecto.
- Requerimientos a probar: tendrá una lista de los requisitos o una referencia al documento "Especificación de requisitos".
- Estrategia de pruebas de aceptación: se describe el flujo de trabajo que se seguirá para la ejecución de las pruebas.
- Evaluación de las pruebas: se describen los criterios de evaluación para las pruebas, como clasificación de las no conformidades, pedidos de cambios y listas de chequeo.
- Cronograma: cronograma de ejecución de las pruebas, en el que se plasmarán, también, las revisiones y auditorías que se planifican.

Caso de prueba

En la estrategia se obtendrán 3 tipos de casos de pruebas distintos: para pruebas unitarias y funcionales, para carga y stress y para las de seguridad.

En el caso de las unitarias, funcionales, carga y stress la plantilla abordará los siguientes aspectos:

- Nombre de proyecto.
- Nombre del producto y versión actual del mismo.
- Nombre del caso de prueba (debe coincidir con el caso de uso referente) y versión actualizada del mismo.
- Control de versiones: tabla que se actualizará con cada cambio que se realice en el documento.
- Descripción general: descripción general del comportamiento del caso de uso al que se hace referencia.
- Condiciones de ejecución: precondiciones que debe cumplir el caso de uso para ejecutarse.
- Secciones: tabla que contendrá las secciones determinadas, de cada una se registrará el nombre, los escenarios que posee, la descripción de la funcionalidad y el flujo central de eventos. Para cada sección los escenarios van a ser flujo básico + flujos alternos.
- Registro de defectos y dificultades detectadas: en esta parte se hace una referencia a la plantilla "No conformidades" referente a este caso de prueba.

En el caso de las pruebas de seguridad, el artefacto se confeccionará teniendo en cuenta los elementos que a continuación se listan:

- Nombre de proyecto.
- Nombre del producto y versión actual del mismo.
- Nombre del caso de prueba y versión actualizada del mismo.
- Control de versiones: tabla que se actualizará con cada cambio que se realice en el documento.
- Descripción general: descripción general del comportamiento del caso de uso al que se hace referencia.
- Condiciones de ejecución: precondiciones que debe cumplir el caso de uso para ejecutarse.
- Funcionalidades a probar para cada rol: donde se describe el comportamiento del sistema ante cada uno de los roles que intervienen en el módulo.
- Registro de defectos y dificultades detectadas: en esta parte se hace una referencia a la plantilla “No conformidades” referente a este caso de prueba.

Defecto de prueba o no conformidad:

Este artefacto será confeccionado por el probador y estará compuesto de la siguiente manera:

- Nombre del proyecto.
- Nombre del producto.
- Versión actualizada del documento.
- Control de versiones: tabla que se actualizará con cada cambio que se realice en el documento.
- Descripción general: descripción de 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: descripción general o lista de los elementos probados, y otros aspectos importantes a tener en cuenta a la hora analizar las no conformidades detectadas.
- Elementos no probados y causas: descripción de 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.
- Registro de defectos y dificultades detectados: tabla que contendrá el elemento que se prueba (documentación o aplicación), número de la no conformidad, descripción de la no conformidad de forma clara, descripción del aspecto correspondiente (escenario en el que se encuentra la deficiencia), etapa de detección, clasificación de la no conformidad, estado de la no conformidad y respuesta del equipo de desarrollo.

- Anexos: apoyo visual que se crea necesario para hacer más entendible la no conformidad detectada.

Los aspectos que registra este artefacto también pueden ser controlados con la utilización del Redmine. En el caso de esta propuesta se archivarán en la plantilla según el expediente que utiliza el proyecto FTC.

Registro de prueba unitaria o integración

Esta plantilla registra los resultados de las pruebas unitarias y de integración realizada, para ello recogerá los siguientes elementos:

- Nombre del proyecto.
- Nombre del producto.
- Versión actualizada del documento.
- Introducción: dentro de este acápite se describirá el alcance del documento, las definiciones, acrónimos y abreviaturas que se utilizan y las referencias hechas a otros documentos.
- Registro de prueba unitaria o integración: este acápite recoge los detalles de las pruebas como el tipo de prueba (unidad o integración), nombre de la prueba, estado, tipo, última ejecución, ejecutado por, verificado por, descripción de la prueba, criterio de entrada, datos de aceptación y resultado.

Aunque no es uno de los artefactos solicitados según RUP, este documento es solicitado por el expediente de proyecto 2.0 y quedará plasmado debido a la política de la UCI de documentar todo el trabajo de un proyecto en un expediente que estandarice los resultados.

Evaluación de la prueba:

Este documento se redactará luego de finalizado cada nivel de prueba, donde el probador comparará el resultado obtenido con el resultado esperado y arribará a conclusiones en conjunto con el resto del equipo de pruebas. Este documento estará conformado por los siguientes elementos:

- Nombre del proyecto.
- Nombre del producto.
- Versión actualizada del documento.
- Control de versiones: tabla que se actualizará con cada cambio que se realice en el documento.
- Introducción: dentro de este acápite se describirá el alcance del documento y las referencias hechas a otros documentos.

- Recursos: se refleja el escenario de prueba en cuestión. Se listan, también, las herramientas y los artefactos que se utilizaron como apoyo.
- Resumen del estado del artefacto: en este acápite se reflejan las no conformidades encontradas en las pruebas en cuestión, especificando la cantidad de cada tipo. Además, se realiza un cronograma de los artefactos que se generarán para dicha prueba.

2.5. Descripción de la aplicación de las pruebas

El primer paso que se debe llevar a cabo es la elaboración del "Plan de pruebas" del proyecto, donde el administrador de pruebas analizará, según las características del producto, las pruebas que se le aplicarán al sistema para comprobar que cumpla con las especificaciones planteadas por el cliente. En este documento queda plasmado el cronograma que guiará todo el proceso con las fechas en que se aplicarán las distintas pruebas y el rol responsable de desempeñar la actividad.

Luego se desarrollarán los cuatro niveles de pruebas según RUP, aplicando las pruebas seleccionadas para la estrategia. El primer nivel que se llevará a cabo será el de pruebas unitarias.

2.5.1. Descripción de la aplicación de pruebas de unidad

Para este nivel deberían ejecutarse pruebas de caja blanca y caja negra, pero la estrategia no contempla las primeras, pues estas se aplican en momentos cercanos a la implementación ya que van a evaluar el código, y teniendo en cuenta que el proyecto se encuentra en una etapa muy avanzada de desarrollo, la aplicación de estas pruebas sólo acarrearía la evaluación nuevamente del código, lo cual sería muy costoso en tiempo, con el cual realmente no se cuenta. De ahí, que las pruebas a implementarse sean solo de caja negra, aunque es importante señalar que es recomendable aplicar pruebas de caja blanca cuando se cuenta con las condiciones para ello, pues con ellas se garantiza que el código sea óptimo.

Por su parte, el primer paso para la aplicación de las pruebas de caja negra sería que luego de terminada la implementación del módulo, el diseñador de prueba elabora un caso de prueba para cada caso de uso, definiendo las secciones y los escenarios a probar.

A partir de los distintos valores, tanto válidos como inválidos, que pueden ser pasados a la aplicación y que se plasmaron en el caso de prueba, el probador evaluará el funcionamiento del sistema y la correspondencia con lo descrito por el diseñador de prueba, lo cual hará de forma manual.

Las no conformidades detectadas en los casos de pruebas se registrarán en el documento "No conformidades" dejando así evidencia de las deficiencias que fueron encontradas. Para garantizar que

se resuelvan los problemas encontrados se hará una prueba exploratoria y tantas iteraciones más como sean necesarias, luego de las cuales deben quedar respondidas las no conformidades, o al menos la mayor parte de ellas, garantizando que, terminadas las pruebas unitarias, el módulo en cuestión esté listo para llevar a cabo la prueba de integración y se encuentre libre de defectos. Todo este procedimiento se repetirá para cada uno de los módulos.

Terminado este nivel de prueba, se confecciona el "Registro de la pruebas unitarias o integración", donde se recogen los resultados arrojados de manera general, por todos los módulos.

2.5.2. Descripción de la aplicación de pruebas de integración

Terminadas las pruebas unitarias se procede a probar que el sistema funcione correctamente con todos los módulos interactuando entre sí. Para ello se desarrollarán pruebas de integración funcionales de manera ascendente, donde se utilizarán, también, casos de pruebas que describan las respuestas que debe dar el sistema. De igual manera, esta prueba se realizará primeramente de forma manual y luego con la utilización del Selenium IDE.

Con la utilización de los casos de pruebas el probador verificará que la aplicación funciona según lo descrito y cumple con los requisitos. La prueba se hará, inicialmente, con la interacción entre dos módulos y luego se irá aumentando el número de componentes hasta llegar a integrar los cinco módulos del proyecto.

Como parte indispensable del proceso se debe plasmar en el documento "No conformidades" las deficiencias detectadas, el cual será utilizado y actualizado en cada iteración que se realice. Para ello se establece que se debe iterar 3 veces, donde, en cada iteración, se comprueba la corrección de los errores anteriormente encontrados y se evalúa la condición actual del sistema.

De ser necesario, se puede realizar otra iteración para garantizar que no existan problemas en el funcionamiento del producto y salga con la calidad esperada.

Terminadas todas las iteraciones, y habiendo corregido todos los defectos detectados, se elabora el documento "Evaluación de prueba de integración" para reflejar los resultados arrojados por este nivel y se da paso así a las pruebas de sistema.

2.5.3. Descripción de la aplicación de pruebas de sistema

Ya probado que los módulos funcionan de manera integrada, se debe probar el funcionamiento como tal del sistema. Para ello, debido al gran número de este tipo de pruebas existentes, se seleccionan las pruebas que se consideren necesarias para garantizar la calidad del producto según lo solicitado por el cliente. En el caso del producto GeForza, el cumplimiento con los requisitos establecidos se puede comprobar realizando pruebas de carga, de seguridad, de usabilidad, de stress

y volumen. Algunas de estas pruebas se pueden realizar de manera automatizada, donde el diseñador de pruebas selecciona la herramienta a utilizarse, y otras se realizan de forma manual.

A continuación se describe cómo realizar cada una de las pruebas.

Prueba de carga:

Para realizar esta prueba el probador, con la ayuda de la herramienta JMeter y la descripción del caso de prueba confeccionado por el diseñador de pruebas, simulará un alto nivel de demanda de procesamiento para comprobar si el sistema puede manejar el volumen de actividades que se le están solicitando en ese momento.

El resultado de esta prueba se registra, de igual manera, en el documento "No conformidades" correspondiente, el cual se actualizará en cada una de las iteraciones que se realicen. Si en la primera revisión al sistema pasa la prueba satisfactoriamente, no se volverá a comprobar esta función; de ocurrir cambios o encontrarse deficiencias sí se continuará iterando tantas veces como sea necesario. El hecho es que, terminada esta prueba, el sistema debe ser capaz de soportar un alto nivel de carga acorde a las necesidades del cliente.

Prueba de *stress*:

Para esta prueba el probador se encargará de chequear la respuesta del sistema ante condiciones anormales que se puedan presentar, para ello utilizará el JMeter para simular estas situaciones, apoyándose, además, en la descripción del caso de prueba elaborado por el diseñador.

Según transcurre la prueba, si se encuentran deficiencias, se registrarán los resultados en el documento de "No conformidades" de igual manera que en las pruebas anteriores, lo que traerá consigo que se realicen otras iteraciones hasta que se limen las incorrecciones localizadas. De resultar exitosa, no será necesaria una nueva iteración.

Prueba de volumen:

En este caso el probador verificará el comportamiento de la aplicación y la Base de Datos con volúmenes de datos almacenados similares a los esperados en la explotación real del sistema. Para esto utilizará el Data Generator for PostgreSQL para hacer las simulaciones de los pedidos.

Esta, como el resto de las pruebas, registrará las deficiencias detectadas en el documento "No conformidades". Esta prueba se aplicará sólo una vez; si se producen cambios al sistema, entonces sí se volverá a realizar y se actualizará la versión del documento de "No conformidades", registrándose los nuevos resultados.

Prueba de seguridad:

El probador verificará, haciendo uso del caso de prueba descrito para esta funcionalidad por el diseñador de pruebas, el nivel de eficiencia del control de acceso de usuarios de la aplicación, o sea, patentizará que al sistema accede el personal autorizado con los permisos requeridos para cada usuario.

De no existir la seguridad precisa, entonces se lanzará una no conformidad, lo que traerá consigo que se vuelva a ejecutar la prueba para garantizar que fue corregido el error. Sólo se detendrá la comprobación si el sistema pasa la prueba satisfactoriamente.

Esta prueba también puede realizarse de manera automatizada, donde se evaluaría tanto la efectividad del método de encriptamiento de contraseña del sistema como la seguridad de la base de datos, haciendo uso del Shadow Security Scanner o el propio JMeter para la realización de la misma. Es importante destacar que en la aplicación de la estrategia para el presente trabajo de diploma no se contempla esta automatización debido a que se dispone poco tiempo para la misma, no obstante, es recomendable el uso esta prueba, además, se asume que el sistema es seguro debido a que la arquitectura utilizada fue tomada del producto Cedrux, desarrollado por el proyecto ERP, el cual ya está probado y garantiza la existencia de métodos de encriptamiento y otras estrategias de seguridad como el tratamiento de inyecciones SQL, etc.

Prueba de usabilidad:

Aquí, el diseñador de prueba establecerá una lista de chequeo que solicite los aspectos indispensables para que el sistema cumpla con los requisitos de usabilidad que platea este tipo de prueba.

La lista de chequeo debe comprobar que:

- Esté la información libre de errores gramaticales, ortográficos y de los errores tipográficos.
- Los iconos representan las acciones a las que están asociados.
- Existe el menú de la "ayuda".
- Los comandos y las opciones apropiadas están en cada menú.
- En cajas de diálogo "tabuladas", los nombres de las lengüetas o pestañas no son abreviaturas.
- El botón de cancelación se activa cuando se están realizando cambios.
- Un botón de comando debe habilitarse cuando la acción se corresponda con su uso y no en otras ocasiones, el botón está habilitado o no lo está cuando le corresponde.
- Los botones de comando son de tamaño y forma similares.
- El doble click en la barra de "Título" es equivalente a restaurar y volver a la posición anterior.

- Cuando se abre la aplicación, esta ocupará la posición que ocupaba cuando se cerró anteriormente.
- Cada una de las opciones debe dar las respuestas apropiadas y descritas, según los requisitos, en los CU (caso de uso).
- Cada comando de menú tiene una secuencia *hot-key* la cual lo invocará cuando sea apropiado
- La funcionalidad de la tecla escape es correcta.
- La aplicación cuenta con los mensajes para cuando se haga necesario.
- El texto de los mensajes se corresponde con lo que se está abordando.
- La aplicación cuenta con un ambiente amigable al interactuar.
- La aplicación cuenta con la seguridad necesaria.
- La aplicación tiene definido un conjunto de roles al interactuar.

Todos estos aspectos irán encaminados a determinar cómo utilizan los usuarios el sistema al procesar datos o preparar informes, además, el mismo debe ser amigable y de fácil operación, con guía interactiva de orientación al usuario y mensajes indicativos de ocurrencias del sistema.

El probador controla y registra, en la propia plantilla de la lista de chequeo, la correspondencia del sistema con los aspectos señalados. Encontrada la inexistencia de alguno de estos elementos pedidos, se plantea en el documento de "No conformidades". Pasado un tiempo se volverá a ejecutar la revisión para comprobar que se agregaron los elementos que no se encontraron anteriormente o corregir errores detectados, se esa manera se aplicarán las iteraciones hasta eliminar las inconformidades.

Cuando ya han sido probadas las capacidades del sistema se confecciona el documento "Evaluación de la prueba de sistema", donde se evalúan los resultados proyectados en este nivel; luego de esto, ya el producto estará listo para someterse a las pruebas de aceptación.

2.5.4. Descripción de la aplicación de la prueba de aceptación

Una vez finalizadas todas las pruebas anteriores, y estando seguros de que el sistema funciona correctamente y con un número reducido de defectos, se hace necesario que el software cuente con la aprobación del cliente, para saber si este, en realidad, cumple con los requisitos especificados, pues aunque el sistema tenga un número reducido de defectos, puede que no sea lo que necesita realmente el cliente.

Para ello se confecciona, primeramente, el "Plan de pruebas de aceptación" que abarcará los siguientes elementos:

- Nombre del proyecto.

- Nombre del producto.
- Versión actualizada del producto.
- Control de versiones: tabla que se actualizará con cada cambio que se realice en el documento.
- Introducción: dentro de este acápite se describirá el alcance del documento, los objetivos, una descripción del proyecto y el producto y la evolución del plan.
- Recursos: se resumirán los responsables de la aplicación de cada una de las pruebas, así como los escenarios de pruebas y los recursos.
- Estrategia de pruebas de aceptación: contendrá la representación de manera resumida del flujo a seguir, la descripción de las estrategias y tipos de pruebas y los datos de pruebas.
- Documentos generados durante la realización de las pruebas: se relacionan todos aquellos documentos generados durante la realización de las pruebas.
- Evaluación de las pruebas: se describen los criterios de evaluación para las pruebas, como clasificación de las no conformidades y pedidos de cambios.
- Cronograma de trabajo: cronograma de ejecución de las pruebas.
- Anexos.

Establecido el plan, sólo resta elaborar la estrategia de pruebas de aceptación detalladamente. Para esto se debe tener en cuenta que este tipo de prueba abarca en sí misma las pruebas de unidad, integración y sistema, con la diferencia que el probador será el propio cliente.

Para las pruebas de unidad se prepararán, por el equipo de desarrollo, los casos de prueba a seguir y el cliente los utilizará como guía. Si se detecta alguna diferencia entre lo descrito con el funcionamiento real de la aplicación, entonces se lanzarán las no conformidades correspondientes que serán respondidas por el equipo de desarrollo.

Las pruebas de integración se aplicarán, igualmente, guiadas por los casos de pruebas elaborados y se hará un tratamiento de no conformidades de manera similar a la prueba anterior.

Evaluado por el cliente el funcionamiento unitario e integrado de los distintos módulos, se procede a aplicar pruebas de sistema, donde el responsable de realizar dicha prueba válida la conformidad con sus necesidades. De no ser así, se toman las no conformidades y se procede a la corrección de estas.

Terminado todo este proceso, el cliente ya estará en condiciones de establecer concretamente su conformidad con el producto o no, lo cual sería el resultado de este nivel de prueba y de todo el proyecto en sí.

Es importante destacar que en la aplicación de la estrategia para el presente trabajo de diploma no se contempla este tipo de prueba, pues la misma se propone al sistema completo y actualmente no

está terminado en su totalidad, aclarando siempre que una vez probado y liberado el sistema, sí es necesario realizar este nivel de prueba con el cliente.

2.6. Conclusiones

Con el desarrollo de este capítulo, quedó establecida la estrategia de pruebas de software a seguir en el proyecto FTC, con lo que se establece una guía que garantiza la organización del trabajo, ganando en tiempo y asegurando que el producto final tendrá un número reducido de defectos, contribuyendo al aseguramiento de la calidad del mismo.

Capítulo 3: Aplicación de la propuesta

3.1. Introducción

En el presente capítulo se plasmará todo el proceso de aplicación de las pruebas planificadas al producto GeForza, exponiéndose los artefactos que se generaron durante todo el proceso y los resultados arrojados. Además, se evalúan dichos resultados para saber el éxito de la implementación de las pruebas y de la aplicación de la estrategia.

3.2. Establecimiento del "Plan de pruebas"

El primer paso es el establecimiento del "Plan de pruebas", ya que será el artefacto que guiará todo el proceso. En este documento se plasmaron, entre otros, los elementos que a continuación se muestran:

- **Roles y responsabilidades**

Tabla # 2: Roles y responsabilidades del "Plan de pruebas". Fuente: elaboración propia.

Rol	***Cantidad***	***Responsabilidad***
Administrador de calidad.	*1*	- *Asegurar la calidad en el proceso de desarrollo de software del proyecto productivo.* - *Asegurar que la aplicación producida se ajusta a las especificaciones y esté razonablemente libre de errores.* - *Proporcionar una metodología para realizar las pruebas.*
Diseñador de pruebas.	*1*	- *Es responsable de identificar las técnicas y herramientas necesarias.* - *Define y estructura los elementos de prueba.* - *Diseñar los casos de prueba.*
Probador.	*2*	- *Controlar la calidad de los diseños de casos de pruebas realizados en los proyectos.* - *Diseñar casos de prueba.* - *Evaluar y documentar el resultado de las pruebas realizadas al software.*

- **Despliegue del sistema**

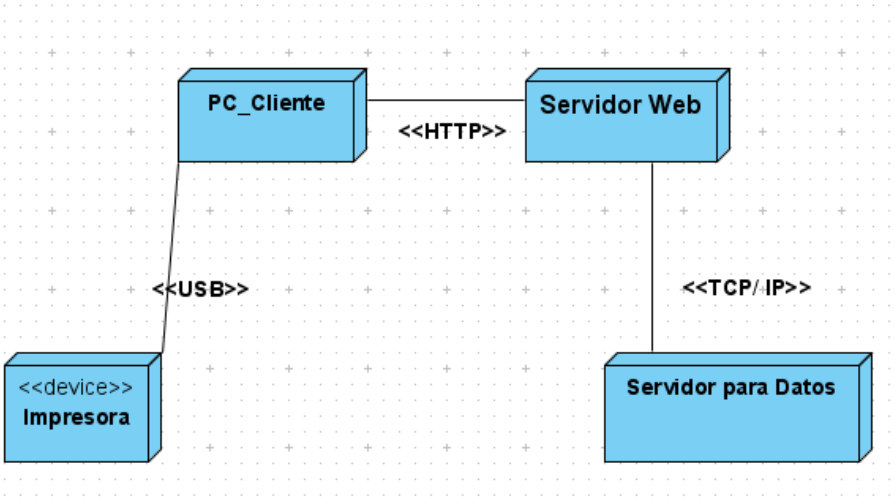

Fig. 4: Despliegue del sistema recogido en el "Plan de pruebas". Fuente: "Plan de pruebas".

- **Recursos del sistema**

Tabla # 3: Recursos del sistema. Fuente: elaboración propia.

Ordenadores Clientes	
Cantidad	2
Descripción	2.20GHz, 0.97 GB de RAM, Intel(R) Core(TM) Duo CPU E4500@ 2.20GHz
Software base	Sistema operativo Windows XP Service Pack 2, JMeter, Selenium IDE, Data Generator for PostgreSQL.

- **Servidores**

Tabla # 4: Servidores. Fuente: elaboración propia.

Recurso	***Tipo***
Servidor de Base de Datos (Esdras)	*Xeon a 3.0 GHz 1 GB de RAM y 2 discos duros de 36 y 250 GB*
Servidor Web (Adonai)	*Xeon a 3.0 GHz 1 GB de RAM y 3 discos duros de 36 GB cada uno con RAID 5*

- **Requisitos a probar**

Se hace referencia al documento "Especificación de requisitos".

En el Anexo 19 y Anexo 20 se relacionan los requisitos funcionales y no funcionales de los módulos "Sistema Matriz" y "Administración".

- **Estrategia de pruebas de aceptación**

Para la realización de las pruebas de aceptación se harán, primero, pruebas unitarias para comprobar el funcionamiento de cada uno de los módulos que integran el proyecto. Luego se harán pruebas de integración para comprobar el funcionamiento integrado del sistema. Seguidamente se le aplicarán pruebas de sistema, dentro de estas se escogieron pruebas de carga, *stress*, usabilidad, volumen y seguridad, las cuales se realizarán de forma manual y algunas, además, se harán de forma automatizada con la utilización de la herramienta JMeter. Luego de todo esto sólo resta la implementación de las pruebas de aceptación donde el propio cliente dará la aprobación del producto.

3.3. Aplicación de las pruebas

Ya definido el documento "Plan de pruebas", se procede a probar el producto, para ello se implementan los cuatro niveles de pruebas definidos con los artefactos obtenidos en cada uno. Esto se le aplica a los módulos "Administración" y "Sistema Matriz", seleccionados como muestra para la validación de la estrategia en este trabajo de diploma.

Seguidamente se describen las pruebas realizadas y los resultados arrojados.

3.3.1. Pruebas unitarias

Para llevar a cabo las pruebas de unidad se realizaron las descripciones de casos de pruebas, con la plantilla establecida, para cada uno de los casos de uso críticos de la muestra, obteniendo un total de 17 casos de pruebas descritos para los módulos "Administración" y "Sistema Matriz".

Seguidamente, se muestran algunos aspectos recogidos en el documento "DCP_Gestionar Carrera", del módulo "Administración", tomado como caso de estudio para exponer el trabajo realizado en este aspecto.

- Descripción general

El caso de uso comienza cuando el Administrador selecciona la opción Gestionar Carrera. A partir de ahí tiene la opción de Adicionar, Modificar, Eliminar, Buscar o Exportar. Posteriormente realiza la acción seleccionada referente a la carrera deseada.

- Condiciones de ejecución

Se debe estar autenticado en el sistema con el rol de administrador.

- Secciones

Tabla # 5: Secciones a probar del documento "DCP_Gestionar carrera". Fuente: elaboración propia.

Nombre de la sección	***Escenarios de la sección***	***Descripción de la funcionalidad***	***Flujo central***
SC1: Adicionar carrera.	*EC 1.1: Adicionar una carrera correctamente.*	*Se adiciona una nueva carrera al sistema y se*	*-Se selecciona la opción "Adicionar".* *- Llenar todos los campos*

		muestra un mensaje de confirmación.	*- Seleccionar "Aplicar".* *- Seleccionar "Aceptar".* *- Muestra un mensaje indicando que el organismo fue creado con éxito.*
	EC 1.2: Adicionar una carrera pasando código existente.	*No se adiciona carrera al sistema.*	*-Se selecciona la opción "Adicionar".* *- Llenar los campos pasando código existente.* *- Seleccionar "Aplicar".* *.- Muestra un mensaje de error indicando que el código ya existe.* *-Seleccionar "Aceptar"*
	EC 1.3: Adicionar una carrera dejando datos en blanco.	*No se adiciona carrera al sistema.*	*-Se selecciona la opción "Adicionar".* *- Llenar los campos dejando datos en blanco.* *- Seleccionar "Aplicar".* *-Se muestra un mensaje indicando que hay campos incorrectos.* *-Seleccionar "Aceptar"*
	EC 1.4: Cancelar la operación	*No se adiciona carrera al sistema.*	*-Se selecciona la opción "Adicionar".* *- Llenar los campos.* *- Seleccionar "Cancelar".*
SC2: Modificar carrera.	*EC 2.1: Modificar carrera correctamente.*	*Se modifica una carrera en el sistema y se muestra un mensaje de confirmación.*	*-Se selecciona una carrera.* *-Se selecciona la opción "Modificar"* *- Se cambian los datos deseados.* *- Seleccionar ¨Aceptar¨.* *- Muestra un mensaje indicando que la carrera fue modificada con éxito.*
	EC 2.2: Modificar carrera pasando código existente.	*No se modifica la carrera al sistema.*	*-Se escoge una carrera.* *-Se selecciona la opción "Modificar ".* *- Se cambian los datos deseados pasando código existente* *-Seleccionar ¨Aceptar¨.* *-Se muestra un mensaje indicando*

			que el código está siendo utilizado. *-Seleccionar "Aceptar".*
	EC 2.3: Cancelar operación.	*No se modifica carrera al sistema.*	*-Se escoge una carrera.* *-Se selecciona la opción "Modificar".* *- Se cambian los datos deseados.* *- Seleccionar "Cancelar".*
SC3: Eliminar carrera.	*EC 3.1: Eliminar carrera correctamente.*	*Se elimina una carrera al sistema y se muestra un mensaje de confirmación.*	*-Se selecciona una carrera.* *-Se selecciona la opción "Eliminar".* *-Muestra un mensaje de confirmación:" Seguro que desea eliminar la carrera".* *-Seleccionar la opción "Aceptar".* *-Muestra un mensaje indicando que se eliminó la carrera.*
	EC 3.2: Cancelar la operación.	*No se elimina una carrera al sistema.*	*-Se selecciona una carrera.* *-Se selecciona la opción "Eliminar".* *-Muestra un mensaje de confirmación:" Seguro que desea eliminar la carrera".* *-Se selecciona la opción "Cancelar".*
SC4: Buscar carrera.	*EC 4.1: Buscar carrera correctamente.*	*Se busca una carrera en el sistema.*	*- Se selecciona la opción Buscar.* *- Introduce uno o varios criterios de búsqueda.* *-Selecciona "Buscar".* *-Se muestran los datos de la(s) carrera(s) buscada(s).*
	EC 4.2: Buscar carrera pasando código inexistente.	*No se busca carrera en el sistema.*	*- Se selecciona la opción Buscar.* *- Se introduce el criterio.* *-Selecciona "Buscar".* *- Muestra un mensaje indicando que el criterio de búsqueda*

			introducido no genera resultado. *- Seleccionar "Aceptar".*
	EC 4.3: Buscar carrera sin pasar criterio de búsqueda.	*No se busca carrera en el sistema.*	*- Se selecciona la opción Buscar.* *-Selecciona "Buscar".* *- Muestra un mensaje indicando que no se puede buscar.* *- Seleccionar "Aceptar".*
	EC 4.4: Cancelar la búsqueda.	*No se busca una carrera en el sistema.*	*- Se selecciona la opción Buscar.* *- Introduce uno o varios criterios de búsqueda.* *-Selecciona "Cancelar".*
	EC 4.5: Limpiar búsqueda.	*Se limpian los datos resultantes de la búsqueda.*	*- Se selecciona la opción Buscar.* *- Introduce uno o varios criterios de búsqueda.* *-Selecciona "Buscar".* *-Se muestran los datos de la(s) carrera(s) buscada(s).* *-Seleccionar la opción" Limpiar búsqueda".* *-Vuelve a la ventana principal de Gestionar carrera mostrando todas las carreras existentes.*

SC1: Adicionar carrera

Tabla # 6: Escenarios de la sección Adicionar carrera. Fuente: elaboración propia.

ID del escenario	***Escenario***	***Código***	***Carrera***	***Nivel Educacional***	***Familia de especialidades***	***Rama de ciencia***	***Respuesta del sistema***	***Resultado de la prueba***
EC 1.1	*Adicionar una carrera correctamente.*	*V(12365987)*	*V (Ética)*	*V(Nivel Superior)*	*NA*	*V(Ciencias sociales y Humanitarias)*	*Muestra el mensaje: "Se ha insertado la carrera satisfactoria*	*Se adiciona una carrera al sistema.*

		V(12364987)	V (Economía)	V(Técnico Medio	V(Economía)	NA	mente"	
EC 1.2	Adicionar una carrera pasando código existente.	I(12345623)	V (Economía)	V(Técnico Medio)	V(Economía)	NA	Muestra el mensaje de error : "No se puede realizar la operación deseada, ese id de la carrera ya está en uso en la base de datos".	No se adiciona carrera al sistema.
EC 1.3	Adicionar una carrera dejando datos en blanco.	I(vacío)	V (Economía)	V(Técnico Medio)	V(Economía)	NA	Muestra un mensaje de error : "Datos incorrectos".	No se adiciona una carrera al sistema.
		V(12345623)	I(vacío)	V(Técnico Medio)	I(vacío)	NA		
		V(12345623)	V (Economía)	I(vacío)	NA	I(vacío)		
EC 1.4	Cancelar la operación.	NA	NA	NA	NA	NA	Vuelve a la ventana principal para Gestionar carrera.	Se cancela la operación y vuelve a la ventana anterior.

SC2: Modificar carrera

Tabla # 7: Escenarios de la sección Modificar carrera. Fuente: elaboración propia.

ID del escenario	Escenario	Código	Carrera	Nivel Educacional	Familia de especialidades	Rama de ciencia	Respuesta del sistema	Resultado de la prueba
EC 2.1	Modificar carrera correctament	V(12394685)	V (Ética)	V(Nivel Superi	NA	V(Ciencias sociales	Muestra el mensaje: "Se ha	Se modifica una carrera en el

	e.			or)		y Humanitarias)	modificado satisfactoriamente la carrera".	sistema.
		V(12364987)	V (Ética)	V(Técnico Medio)	V(Especialidades formadas por el MINCULT)	NA		
EC 2.2	Modificar una carrera pasando código existente.	I(12345623)	V (Ética)	V(Técnico Medio)	V(Especialidades formadas por el MINCULT)	NA	Muestra el mensaje de error : "No se puede realizar la operación deseada, ese id de la carrera ya está en uso en la base de datos".	No se modifica carrera al sistema.
EC 2.3	Cancelar la operación.	NA	NA	NA	NA	NA	Vuelve a la ventana anterior.	Se cancela la operación y vuelve a la ventana anterior.

SC3: Eliminar carrera

Tabla # 8: Escenarios de la sección Eliminar carrera. Fuente: elaboración propia.

ID del escenario	Escenario	Código	Carrera	Nivel Educacional	Familia de especialidades	Rama de ciencia	Respuesta del sistema	Resultado de la prueba
EC 3.1	Eliminar carrera correctamente.	NA	NA	NA	NA	NA	Muestra un mensaje: "Se ha eliminado satisfactoria	Se elimina la carrera del sistema.

							mente la carrera".	
EC 3.2	*Cancelar la operación.*	*NA*	*NA*	*NA*	*NA*	*NA*	*Vuelve a la ventana anterior.*	*No se elimina carrera del sistema.*

SC4: Buscar carrera

Tabla # 9: Escenarios de la sección Buscar carrera. Fuente: elaboración propia.

ID del escenario	Escenario	Código	Carrera	Nivel Educacional	Familia de especialidades	Rama de ciencia	Respuesta del sistema	Resultado de la prueba
EC 4.1	*Buscar carrera correctamente.*	*V(vacío)*	*V (Agronomía)*	*V(vacío)*	*V(vacío)*	*V(vacío)*	*Muestra los datos de la(s) carrera(s) resultante(s).*	*Se busca una carrera en el sistema.*
		V(45789633)	*V (Agronomía)*	*V(vacío)*	*V(vacío)*	*V(vacío)*		
		V(vacío)	*V(vacío)*	*V(Técnico Medio)*	*V(vacío)*	*V(vacío)*		
		V(vacío)	*V(vacío)*	*V(vacío)*	*V(Especialidades formadas por el MINCULT)*	*V(vacío)*		
		V(vacío)	*V(vacío)*	*V(vacío)*	*V(vacío)*	*V(Ciencias sociales y Humanitarias)*		
EC 4.2	*Buscar carrera pasando código inexistente.*	*I(15874693)*	*V(vacío)*	*V(vacío)*	*V(vacío)*	*V(vacío)*	*Muestra el mensaje: "No existe el código 15874693"*	*No se encuentra el resultado de la búsqueda.*
EC 4.3	*Buscar carrera sin pasar criterio de búsqueda.*	*I(vacío)*	*I(vacío)*	*I(vacío)*	*I(vacío)*	*I(vacío)*	*Muestra el mensaje: "Debe seleccionar*	*No se realiza la búsqueda.*

							al menos un campo".	
EC 4.4	*Cancelar la operación.*	*NA*	*NA*	*NA*	*NA*	*NA*	*Vuelve a la ventana anterior.*	*No se busca carrera en el sistema.*
EC 4.5	*Limpiar la búsqueda.*	*NA*	*NA*	*NA*	*NA*	*NA*	*Borra el resultado de la búsqueda actual y muestra todas las carreras existentes.*	*Borra el resultado de la búsqueda actual y muestra todas las carreras existentes.*

El resto de las descripciones de casos de pruebas pueden citarse en Anexo 9 y Anexo 10.

Al aplicarse las pruebas, se detectan deficiencias en el sistema para ese caso de prueba, de ahí que se genere el documento "NC_DCP_Gestionar carrera" donde se archivan los defectos detectados en cada iteración, en este caso se presentan las no conformidades de la prueba exploratoria y la primera iteración, donde RA y PD significan "Resuelta" y "Pendiente" respectivamente, como sigue:

Tabla # 10: No conformidades del documento "NC_DCP_Gestionar carrera". Fuente: elaboración propia.

Elemento	***No***	***No conformidad***	**Aspecto correspondiente**	**Etapa de detección**	**Clasificación**	**Estado NC**	**Respuesta del Equipo Desarrollo**
Aplicación	*1*	*El sistema permite adicionar pasando código existente.*	*SC1: Adicionar carrera. EC 1.2: Adicionar una carrera pasando código existente.*	*Prueba exploratoria.*	*Significativa.*	*RA*	
Aplicación	*2*	*El sistema permite buscar sin pasar ningún criterio de búsqueda y muestra todas las entidades.*	*SC4: Buscar carrera.*	*Primera iteración.*	*Recomendación.*	*PD*	

		Debería lanzar un mensaje de error.					
Aplicación	*3*	*Existen errores ortográficos en los datos existentes.*	*Aplicación.*	*Primera iteración.*	*No Significativa.*	*PD*	
Aplicación	*4*	*En el mensaje de error para adicionar con código existente hay errores ortográficos.*	*SC1: Adicionar carrera- EC 1.2: Adicionar una carrera pasando código existente.*	*Primera iteración.*	*No Significativa.*	*PD*	

El resto de las tablas de no conformidades se encuentran publicadas en el Anexo 14 y Anexo 15.

En general, para los dos módulos seleccionados como muestra, se obtuvo un total de 31 no conformidades en la prueba exploratoria, 27 en la primera iteración y 6 en la tercera iteración. Los resultados de las pruebas unitarias quedaron como lo expresa el siguiente gráfico:

Fig. 5: Resultados de las pruebas unitarias para los módulos Administración y Sistema Matriz. Fuente: elaboración propia.

3.3.2. Pruebas de integración

Luego de probadas las unidades, se procede a comprobar el funcionamiento integrado, para lo cual también se establecen las descripciones de los casos de pruebas funcionales. Para presentar estas descripciones se exponen aspectos del documento “DCP_Gestionar existencia de empleados”, del módulo “Sistema Matriz”, como sigue:

- Descripción general

El caso de uso comienza cuando el Funcionario selecciona la opción Gestionar existencia de empleados. A partir de ahí tiene la opción de Adicionar, Modificar, Eliminar, Buscar o Exportar. Posteriormente realiza la acción seleccionada referente al empleado deseado.

- Condiciones de ejecución

Se debe estar autenticado en el sistema con el rol de funcionario.

- Secciones

Tabla # 11: Secciones aprobar del documento "DCP_Gestionar existencia de empleados". Fuente: elaboración propia.

Nombre de la sección	***Escenarios de la sección***	***Descripción de la funcionalidad***	***Flujo central***
SC1: Adicionar.	*EC 1.1: Adicionar existencia correctamente pulsando el botón Aplicar.*	*Se adiciona una existencia al sistema y se muestra un mensaje de confirmación.*	*-Se selecciona la opción "Adicionar".* *- Llena los campos.* *- Seleccionar "Aplicar".* *- Muestra un mensaje informando que se ha insertado satisfactoriamente mantiene el organismo y la entidad, limpia el resto de los campos dando la opción de volver a adicionar.* *- Seleccionar "Aceptar".*
	EC 1.2: Adicionar existencia correctamente pulsando el botón Aceptar.	*Se adiciona una existencia al sistema y se muestra un mensaje de confirmación.*	*-Se selecciona la opción "Adicionar".* *- Llena los campos.* *- Seleccionar "Aceptar".* *- Muestra un mensaje informando que se ha insertado satisfactoriamente.* *- Seleccionar "Aceptar".*
	EC 1.3: Adicionar existencia dejando campos en blanco.	*No se adiciona una existencia al sistema.*	*-Se selecciona la opción "Adicionar".* *- Llena los campos.* *- Seleccionar "Aplicar" o "Aceptar".* *- Muestra un mensaje de error notificando al usuario que dejó algún campo vacío.* *- Seleccionar "Aceptar".*

	EC 1.4: Adicionar existencia pasando datos existentes.	*No se adiciona una existencia al sistema.*	*-Se selecciona la opción "Adicionar".* *- Llena los campos.* *- Seleccionar "Aplicar" o "Aceptar".* *- Muestra un mensaje indicando que ya existe.* *- Seleccionar "Aceptar".*
	EC 1.5: Cancelar la operación.	*No se adiciona existencia al sistema.*	*-Se selecciona la opción "Adicionar".* *- Llena los campos.* *-Selecciona "Cancelar".* *-Se cierra la ventana para adicionar.*
SC2: Modificar.	*EC 2.1: Modificar existencia correctamente.*	*Se modifica una existencia al sistema y se muestra un mensaje de confirmación.*	*-Se selecciona una de las existencias.* *-Se selecciona la opción "Modificar".* *- Llena los campos a modificar.* *- Seleccionar "Aceptar".* *- Muestra el mensaje "Se ha modificado satisfactoriamente".* *- Seleccionar "Aceptar".*
	EC 2.2: Modificar existencia pasando datos existentes.	*No se modifica una existencia al sistema.*	*- Se selecciona una de las existencias.* *-Se selecciona la opción "Modificar".* *- Llena los campos a modificar.* *- Seleccionar "Aceptar".* *- Muestra un mensaje indicando que no se ha modificado.*
	EC 2.3: Cancelar operación.	*No se modifica una existencia al sistema.*	*- Se selecciona una de las existencias.* *-Se selecciona la opción "Modificar".*

			- Seleccionar “Cancelar”. -Se cierra la ventana para modificar.
SC3: Eliminar.	EC 3.1: Eliminar existencia correctamente.	Se elimina una existencia del sistema y se muestra un mensaje de confirmación.	- Se selecciona una de las existencias. -Se selecciona la opción “Eliminar”. - Muestra el mensaje “Seguro que desea eliminar la existencia”. - Seleccionar “Aceptar”. - Muestra el mensaje “Se ha eliminado satisfactoriamente”. - Seleccionar “Aceptar”.
	EC 3.2: Cancelar la operación.	No se elimina una existencia del sistema.	-Se selecciona una de las existencias. -Se selecciona la opción “Eliminar”. - Muestra el mensaje “Seguro que desea eliminar la existencia”. - Seleccionar “Cancelar”. -Se cierra la ventana para eliminar.
SC4: Buscar	EC 4.1: Buscar una existencia correctamente.	Se busca una existencia en el sistema.	-Se selecciona la opción “Buscar”. -Se selecciona el(los) criterio(s) de búsqueda deseado(s). - Se selecciona la opción “Aceptar”. - Se muestran todas las existencias pertenecientes al año con que se está trabajando y correspondiente a la selección hecha.
	EC 4.2: Buscar sin pasar criterio de búsqueda.	No se busca existencia en el sistema.	-Se selecciona la opción “Buscar”. - Seleccionar “Aceptar”. - Muestra un mensaje indicando que debe seleccionar al menos un campo. - Seleccionar “Aceptar”.

	EC 4.3: Buscar sin existir disponibilidades correspondientes.	*No se busca existencia en el sistema.*	*-Se selecciona la opción "Buscar".* *-Se selecciona el(los) criterio(s) de búsqueda deseado(s).* *- Muestra un mensaje notificando que no hay resultados para mostrar.* *- Seleccionar "Aceptar".*
	EC 4.4: Cancelar la operación.	*No se busca existencia en el sistema.*	*-Se selecciona la opción "Buscar".* *-Se selecciona el(los) criterio(s) de búsqueda deseado(s).* *- Se selecciona la opción "Cancelar".* *- Se cierra la ventana para buscar.*
	EC 4.5: Limpiar Búsqueda.	*Se limpian los datos introducidos para la búsqueda actual.*	*-Se selecciona la opción "Buscar".* *-Se selecciona el(los) criterio(s) de búsqueda deseado(s).* *- Se selecciona la opción "Aceptar".* *- Se muestran todas las existencias en el año con que se está trabajando y correspondiente a la selección hecha.* *-Seleccionar la opción" Limpiar búsqueda".* *-Vuelve a la ventana principal para Gestionar existencia de empleados mostrando todas las disponibilidades existentes.*
SC 5: Exportar	*EC 5.1: Exportar.*	*Se crea un documento pdf.*	*-Seleccionar la opción" Exportar".* *- Crea un documento pdf con los datos seleccionados por el funcionario, teniendo habilitadas todas las opciones que brinda el pdf.*

SC6: Año	*EC 6.1: Mostrar existencias por año.*	*Se muestra una lista de existencias.*	*-Se selecciona un año.* *- Muestra todas las existencias del año seleccionado.*
	EC 6.2: No mostrar existencias por año.	*No se muestran existencias.*	*-Se selecciona un año.* *- Muestra un mensaje notificando que no hay resultados para mostrar.*

SC1: Adicionar

Tabla # 12: Escenarios de la sección Adicionar. Fuente: elaboración propia.

ID del escenario	***Escenario***	***Organismo***	***Entidad***	***Carrera***	***Carreras Asociadas***	***Provincia***	***Municipio***	***Tipo de Plaza***	***Edad***	***Cantidad de Graduados***	***Respuesta del sistema***	***Resultado de la prueba***
EC 1.1	*Adicionar existencias correctamente pulsando el botón Aplicar.*	*V(Ministerio de la Industria Sidero Mecánica)*	*V (Ministerio de la Industria Sidero Mecánica)*	*V (Informática)*	*V(informática vieja)*	*V(Santiago de Cuba)*	*V(Palma Soriano)*	*V(Adiestramiento laboral con Plaza Fija)*	*V(Menores de 30 años)*	*V(50)*	*Se adiciona una existencia al sistema.*	*Muestra un mensaje informando que se ha insertado satisfactoriamente, limpia los campos dando la opción de adicionar nuevamente.*
EC 1.2	*Adicionar existencias correctamente pulsando el botón Aceptar.*	*V(Ministerio de la Industria Sidero Mecánica)*	*V (Ministerio de la Industria Sidero Mecánica)*	*V (Informática)*	*V(informática vieja)*	*V(Santiago de Cuba)*	*V(Palma Soriano)*	*V(Adiestramiento laboral con Plaza Fija)*	*V(Menores de 30 años)*	*V(50)*	*Se adiciona una existencia al sistema.*	*Muestra un mensaje informando que se ha insertado satisfactoriamente.*

EC 1.3	*Adicionar existencias dejando campos en blanco.*	*V(Ministerio de la Industria Sidero Mecánica)*	*V (Empresa Siderúrgica José Martí)*	*I(Vacío)*	*I(Vacío)*	*I(Vacío)*	*I(Vacío)*	*I(Vacío)*	*I(Vacío)*	*I(Vacío)*	*No se adiciona una existencia al sistema.*	*Muestra un mensaje de error notificando al usuario que dejó algún campo vacío.*
		V(Ministerio de Salud pública)	*I(Vacío)*	*I(Vacío)*	*I(Vacío)*	*V(Santiago de Cuba)*	*V(Santiago de Cuba)*	*I(Vacío)*	*I(Vacío)*	*I(Vacío)*		
		I(Vacío)	*V (Empresa Siderúrgica José Martí)*	*I(Vacío)*	*I(Vacío)*	*V (Ciego de Ávila)*	*V (Ciro Redondo)*	*I(Vacío)*	*I(Vacío)*	*I(Vacío)*		
EC 1.4	*Adicionar existencias pasando datos existentes.*	*V(Ministerio de la Industria Sidero Mecánica)*	*V (Ministerio de la Industria Sidero Mecánica)*	*V (Informática)*	*V(informática vieja)*	*V(Santiago de Cuba)*	*V(Palma Soriano)*	*V(Adiestramiento laboral con Plaza Fija)*	*V(Menores de 30 años)*	*V(50)*	*No se adiciona una existencia al sistema.*	*Muestra un mensaje indicando que ya existe.*
EC 1.5	*Cancelar la operación.*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*No se adiciona una existencia al sistema.*	*Se cierra la ventana para adicionar.*

SC2: Modificar

Tabla # 13: Escenarios de la sección Modificar. Fuente: elaboración propia.

ID del escenario	Escenario	Organismo	Entidad	Carrera	Carreras Asociadas	Provincia	Municipio	Tipo de Plaza	Edad	Cantidad de Graduados	Respuesta del sistema	Resultado de la prueba
EC 2.1	Modificar existencias correctamente.	NA	NA	NA	NA	NA	NA	NA	NA	V(60)	Se modifica una existencia al sistema.	Muestra el mensaje "Se ha modificado satisfactoriamente".
EC 2.2	Modificar existencias pasando datos existentes.	NA	NA	NA	NA	NA	NA	NA	NA	I(60)	No se modifica una existencia al sistema.	Muestra un mensaje indicando que no se ha modificado.
EC 2.3	Cancelar la operación	NA	NA	NA	NA	NA	NA	NA	NA	NA	No se modifica una existencia al sistema.	Se cierra la ventana para modificar.

SC3: Eliminar

Tabla # 14: Escenarios de la sección Eliminar. Fuente: elaboración propia.

ID del escenario	Escenario	Organismo	Entidad	Carrera	Carreras Asociadas	Provincia	Municipio	Tipo de Plaza	Edad	Cantidad de Graduados	Respuesta del sistema	Resultado de la prueba
EC 3.1	Eliminar existencias correcta	NA	NA	NA	NA	NA	NA	NA	NA	NA	Se elimina una existenc	Muestra el mensaje "Se ha eliminado

	mente.										*ia del sistema*	*satisfactoriamente".*
EC 3.2	*Cancelar la operación.*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*No se elimina una existencia del sistema*	*Se cierra la ventana para eliminar.*

SC4: Buscar

Tabla # 15: Escenarios de la sección Buscar. Fuente: elaboración propia.

ID del escenario	*Escenario*	*Organismo*	*Entidad*	*Carrera*	*Carreras Asociadas*	*Provincia*	*Municipio*	*Tipo de Plaza*	*Edad*	*Respuesta del sistema*	*Resultado de la prueba*
EC 4.1	*Buscar una existencia correctamente.*	*V(Ministerio de la Industria Sidero Mecánica)*	*V(Vacío)*	*V (Informática)*	*V(informática vieja)*	*V(Santiago de Cuba)*	*V(Palma Soriano)*	*V(Adiestramiento laboral con Plaza Fija)*	*V(Menores de 30 años)*	*Se busca una existencia en el sistema.*	*Se muestran todas las existencias existentes en el año con que se está trabajando y correspondiente a la selección hecha.*
		V(Vacío)	*V(Vacío)*	*V(Vacío)*	*V(Vacío)*	*V(Santiago de Cuba)*	*V(Palma Soriano)*	*V(Adiestramiento laboral con Plaza Fija)*	*V(Menores de 30 años)*		
		V(Vacío)	*V(Vacío)*	*V(Vacío)*	*V(Vacío)*	*V(Vacío)*	*V(Vacío)*	*V(Vacío)*	*V(Menores de 30 años)*		

EC 4.2	Buscar sin pasar criterio de búsqueda.	I(Vacío)	I(Vacío)	I(Vacío)	I(Vacío)	I(Vacío)	I(Vacío)	I(Vacío)	I(Vacío)	No se busca una existencia en el sistema.	Muestra un mensaje indicando que debe seleccionar al menos un campo.
EC 4.3	Buscar sin existir disponibilidades correspondientes.	V(Ministerio de la Informática y las Comunicaciones)	V (Empresa Siderúrgica José Martí)	I(Vacío)	I(Vacío)	I(Vacío)	I(Vacío)	I(Vacío)	I(Vacío)	No se busca una existencia en el sistema.	Muestra un mensaje notificando que no hay resultados para mostrar.
EC 4.4	Cancelar la operación.	NA	NA	NA	NA	NA	NA	NA	NA	No se busca una existencia en el sistema.	Se cierra la ventana para buscar.
EC 4.5	Limpiar Búsqueda.	NA	NA	NA	NA	NA	NA	NA	NA	Se limpian los datos introducidos para la búsqueda actual.	Vuelve a la ventana principal para Gestionar existencia de empleados mostrando todas las disponibilidades existentes.

SC5: Exportar

Tabla # 16: Escenarios de la sección Exportar. Fuente: elaboración propia.

ID del escenario	Escenario	Organismo	Entidad	Carrera	Carreras Asociadas	Provincia	Municipio	Tipo de Plaza	Edad	Cantidad de Graduados	Respuesta del sistema	Resultado de la prueba

EC 5.1	*Exportar.*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*Se crea un documento pdf.*	*Crea un documento pdf con los datos seleccionados por el funcionario, teniendo habilitadas todas las opciones que brinda el pdf.*

SC6: Año

Tabla # 17: Escenarios de la sección Año. Fuente: elaboración propia.

ID del escenario	*Escenario*	*Organismo*	*Entidad*	*Carrera*	*Carreras Asociadas*	*Provincia*	*Municipio*	*Tipo de Plaza*	*Edad*	*Cantidad de Graduados*	*Respuesta del sistema*	*Resultado de la prueba*
EC 6.1	*Mostrar existencias por año.*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*Se muestra una lista de existencias.*	*Muestra todas las existencias del año seleccionado.*
EC 6.2	*No mostrar existencias por año.*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*No se muestran existencias.*	*Muestra un mensaje notificando que no hay resultados para mostrar.*

La descripción anterior es la base para las pruebas manuales, pero también se realizan de forma automatizada con la utilización del Selenium IDE y el Selenium Core, para ello se crearon grabaciones de las secuencias de pasos que se deben seguir para las pruebas, las que se encuentran

ejemplificadas en el Anexo 11. Es importante aclarar que para utilizar estas herramientas de pruebas los casos de pruebas deben estar descritos detalladamente.

Haciendo uso de las descripciones y grabaciones se comprueba el funcionamiento del sistema, dejando constancia de las no conformidades encontradas. Para este caso de estudio se obtiene el documento "NC_DCP_Gestionar existencia de empleados", donde las deficiencias quedan plasmadas de la siguiente manera:

Tabla # 18: No conformidades del documento "NC_DCP_Gestionar existencia de empleados". Fuente: elaboración propia.

Elemento	*No*	*No conformidad*	*Aspecto correspondiente*	*Etapa de detección*	*Clasificación*	*Estado NC*	*Respuesta del Equipo Desarrollo*
Aplicación	*1*	*Existen datos que no tienen que ver con la información que tiene que estar almacenada (ej: aaaaaa).*	*En todos los escenarios.*	*Primera iteración.*	*Significativa.*	*RA*	
Aplicación	*2*	*Cuando se pulsa el botón aceptar sin pasar la cantidad de graduados se emite un mensaje, pero se pierde la ventana para adicionar, algo que no pasa con el botón aplicar.*	*SC: Adicionar.*	*Primera iteración. .*	*Significativa.*	*RA*	
Aplicación	*3*	*Cuando se pulsa el botón aceptar o aplicar sin pasar la cantidad de graduados se emite un mensaje, pero se ponen en blanco todos los campos, lo que hace que haya que empezar el proceso desde cero.*	*SC: Adicionar*	*Primera iteración. .*	*Recomendación.* *El proceso no ha de ser necesario empezarlo desde cero, sino que se pueda poner el dato que falta y los demás seguirían activos.*	*RA*	

Aplicación	*4*	*Cuando se exporta, los datos que están en el, son los del año actual, aunque el parámetro año, este en otro.*	*SC: Exportar*	*Primera iteración. .*	*Significativa.*	*RA*	
Aplicación	*5*	*Cuando se selecciona la carrera asociada, nunca se queda marcada, se puede adicionar pero ese campo nunca tiene una carrera mostrada.*	*SC: Adicionar*	*Segunda iteración.*	*Significativa.*	*PD*	
Aplicación	*6*	*Existen datos que no tienen que ver con la información que tiene que estar almacenada (ej: ilma).*	*SC: Adicionar*	*Segunda iteración.*	*Significativa.*	*PD*	

Luego de la segunda iteración de las pruebas de integración, se obtienen los resultados que se muestran en la siguiente figura, mostrando un aumento de deficiencias en los módulos de la muestra.

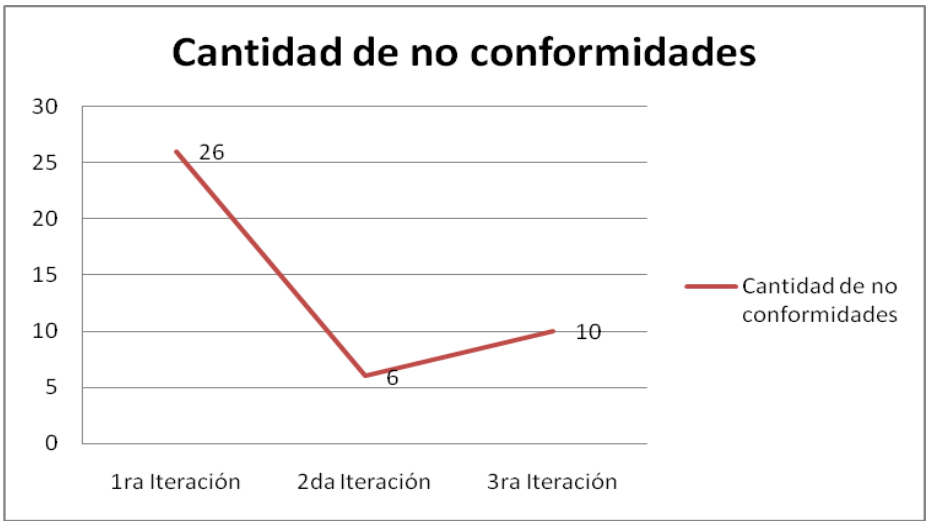

Fig. 6: Resultados de las pruebas de integración para los módulos Administración y Sistema Matriz. Fuente: elaboración propia.

Teniendo estos resultados se procede al siguiente nivel: pruebas de sistema.

3.3.3. Pruebas de sistema

Dentro de este nivel de pruebas, se aplicarán cinco tipos de pruebas: carga, stress, volumen, usabilidad y seguridad. Para cada una de estas pruebas se procede de manera diferente.

3.3.3.1. Pruebas de carga y stress

Para la aplicación de estas pruebas lo primero que se confecciona es el diseño de caso de prueba de carga y stress para cada uno de los módulos, teniendo en cuenta, además que el sistema está hecho para trabajar en ordenadores con bajas prestaciones, por lo que no ha de cumplir grandes requisitos en estos aspectos que se comprueban. Con ese fin se crearon los documentos "DCPCE_Carga_Administración" y "DCPCE_Carga_Matriz", los cuales contienen los aspectos a medir para los módulos "Administración" y "Sistema Matriz" respectivamente. Para un mejor entendimiento, a continuación se presentan las descripciones de las secciones a probar y los resultados proyectados en cada caso según la comprobación con el JMeter se encuentran en el Anexo 11.

Tabla # 19: Secciones a probar para el módulo "Administración". Fuente: elaboración propia.

ID del escenario	Escenarios de la sección	Carga de Trabajo	Descripción	Resultado esperado	Resultado de la prueba
SC 1: Prueba de carga.	*EC 1.1: Prueba de carga para un alto grado de transacciones.*	*50*	*Se evaluará la rapidez de la respuesta del sistema ante un alto grado de transacciones.*	*6 segundos.*	*1 segundo.*
	EC 1.2: Prueba de carga para transacciones con valores de gran magnitud.	*30*	*Se evaluará la rapidez de la respuesta del sistema ante transacciones donde intervengan valores de gran magnitud, como pueden ser valores tomados de otros módulos.*	*6 segundos.*	*0.75 segundos.*
SC 2: Prueba de stress.	*EC 2.1: Prueba de stress ante una sobrecarga*	*125*	*Se evaluará la rapidez de la respuesta del*	*10 segundos.*	*1.4 segundos.*

	de transacciones.		*sistema ante una sobrecarga de transacciones.*		
	EC 2.2: Prueba de stress para transacciones con valores de gran magnitud.	*50*	*Se evaluará la rapidez de la respuesta del sistema ante una sobrecarga de transacciones donde intervengan valores de gran magnitud, como pueden ser valores tomados de otros módulos.*	*12 segundos.*	*1.2 segundos.*

Tabla #20: Secciones a probar para el módulo “Sistema Matriz”. Fuente: elaboración propia.

ID del escenario	***Escenarios de la sección***	***Carga de Trabajo***	***Descripción***	***Resultado esperado***	***Resultado de la prueba***
SC 1: Prueba de carga.	*EC 1.1: Prueba de carga para un alto grado de transacciones.*	*50*	*Se evaluará la rapidez de la respuesta del sistema ante un alto grado de transacciones.*	*6 segundos.*	*1 segundo.*
	EC 1.2: Prueba de carga para transacciones con valores de gran magnitud.	*30*	*Se evaluará la rapidez de la respuesta del sistema ante transacciones donde intervengan valores de gran magnitud, como pueden ser valores tomados de otros módulos.*	*8 segundos.*	*0.75 segundos.*
SC 2: Prueba de stress.	*EC 2.1: Prueba de stress ante una sobrecarga de transacciones.*	*125*	*Se evaluará la rapidez de la respuesta del sistema ante una sobrecarga de transacciones.*	*10 segundos.*	*1.4 segundos.*
	EC 2.2: Prueba de stress para transacciones con valores de	*50*	*Se evaluará la rapidez de la respuesta del sistema ante una*	*11 segundos.*	*1.1 segundos.*

	gran magnitud.		*sobrecarga de transacciones donde intervengan valores de gran magnitud, como pueden ser valores tomados de otros módulos.*		

3.3.3.2.Pruebas de volumen

Al comprobarse las capacidades de la base de datos con el Data Generator for PostgreSQL para un millón de datos, se obtuvo que esta funciona correctamente, donde los resultados completados correctamente se mantuvieron en el rango del 94% al 100% en cada una de las tablas analizadas.

Se detectaron errores de sintaxis e inserción para un 62% de las tablas existentes, aunque en la mayoría de los casos fueron el resultado de que se valida que no se inserten caracteres inválidos.

3.3.3.3.Pruebas de seguridad

Tras la aplicación de esta prueba, quedaron establecidos los diseños de casos de prueba de seguridad para cada uno de los módulos. Seguidamente se ejemplifican las funcionalidades tomadas para ser probadas al módulo “Sistema Matriz”.

Tabla # 21: Funcionalidades a probar para el módulo “Sistema Matriz”. Fuente: elaboración propia.

Escenarios			***Roles***	***Descripción de la funcionalidad a probar***
			Administrador	
EC 1: ***Gestionar existencia de empleados.***	***EC 1.1: Datos mostrados al acceder a Gestionar existencia de empleados.***	***Resultado esperado***	*Se muestran todas las opciones que se pueden realizar permitiendo hacer cambios.*	*Al acceder al CU “Gestionar existencia de empleados”, el sistema sólo le debe mostrar al usuario los datos y servicios a los que tiene acceso.*
		Resultado de la prueba	*Se muestran todas las opciones que se pueden realizar y registra los cambios.*	

EC 2: ***Gestionar Demanda de graduados por carreras.***	***EC 2.1: Datos mostrados al acceder a Gestionar Demanda de graduados por carreras.***	***Resultado esperado***	*Se muestran todas las opciones que se pueden realizar permitiendo hacer cambios.*	*Al acceder al CU "Gestionar demanda de graduados por carreras", el sistema sólo le debe mostrar al usuario los datos y servicios a los que tiene acceso.*
		Resultado de la prueba	*Se muestran todas las opciones que se pueden realizar y registra los cambios.*	
EC 3: ***Gestionar disponibilidad de noveno grado.***	***EC 3.1: Datos mostrados al acceder a Gestionar disponibilidad de noveno grado.***	***Resultado esperado***	*Se muestran todas las opciones que se pueden realizar permitiendo hacer cambios.*	*Al acceder al CU "Gestionar disponibilidad de noveno grado", el sistema sólo le debe mostrar al usuario los datos y servicios a los que tiene acceso.*
		Resultado de la prueba	*Se muestran todas las opciones que se pueden realizar y registra los cambios.*	
EC 4: ***Gestionar disponibilidad de nivel medio.***	***EC 4.1: Datos mostrados al acceder a Gestionar disponibilidad de nivel medio.***	***Resultado esperado***	*Se muestran todas las opciones que se pueden realizar permitiendo hacer cambios.*	*Al acceder al CU "Gestionar disponibilidad de nivel medio", el sistema sólo le debe mostrar al usuario los datos y servicios a los que tiene acceso.*
		Resultado de la prueba	*Se muestran todas las opciones que se pueden realizar y registra los cambios.*	
EC 5: ***Gestionar disponibilidad de nivel superior.***	***EC 5.1: Datos mostrados al acceder a Gestionar disponibilidad de nivel superior.***	***Resultado esperado***	*Se muestran todas las opciones que se pueden realizar permitiendo hacer cambios.*	*Al acceder al CU "Gestionar disponibilidad de nivel superior", el sistema sólo le debe mostrar al usuario los datos y servicios a los que tiene acceso.*
		Resultado de la prueba	*Se muestran todas las opciones que se pueden realizar y registra los cambios.*	

EC 6: ***Gestionar proyección de la demanda.***	***EC 5.1: Datos mostrados al acceder a Gestionar proyección de la demanda.***	***Resultado esperado***	*Se muestran todas las opciones que se pueden realizar permitiendo hacer cambios.*	*Al acceder al CU "Gestionar proyección de la demanda", el sistema sólo le debe mostrar al usuario los datos y servicios a los que tiene acceso.*
		Resultado de la prueba	*Se muestran todas las opciones que se pueden realizar y registra los cambios.*	

Las funcionalidades a probar para el módulo "Administración" se encuentran en el Anexo 13 del trabajo.

Al haber aplicado la prueba se obtiene que el sistema cuenta con una correcta validación para el control de acceso, dando los permisos y privilegios a los usuarios según les corresponde. Con ese resultado se establece que la prueba no cumplió su objetivo de detectar errores, pero se comprueba que el funcionamiento del sistema resulta exitoso.

3.3.3.4.Pruebas de usabilidad

La implementación de esta prueba está guiada por una lista de chequeo establecida para este nivel, la que se encuentra en el Anexo 12, en la que se reúnen los aspectos que se le miden a la aplicación para comprobar que cumple con los requisitos no funcionales.

Al realizarse esta comprobación se obtuvo que de los 19 elementos que se deseaban comprobar, sólo 3 no fueron encontrados, demostrando el cumplimiento, casi total, del producto GeForza con los requisitos no funcionales comprobados para los módulos "Administración" y "Sistema Matriz". Tras los resultados obtenidos se notifica al equipo de desarrollo los requisitos que aún deben ser corregidos.

3.3.4. Pruebas de aceptación

La ejecución de estas pruebas debe estar guiada por un plan de pruebas. En este caso el plan quedó como se plantea a continuación:

- Introducción.

Este documento se confecciona con el objetivo de definir el documento "Plan de pruebas de aceptación" para el Proyecto Fuerza de Trabajo Calificada (FTC) específicamente el producto GeForza. En el documento "Descripción de CUS" se encuentran descritos todos los casos de uso del sistema que soportan los procesos de negocio y que serán sometidos al proceso de aceptación para cada uno de los siguientes módulos:

- Administración
- Sistema Matriz
- Distribución
- Ingreso

Se nombran además el listado de los artefactos de apoyo a las pruebas.

Tabla # 22: Lista de artefactos de apoyo a las pruebas de aceptación. Fuente: elaboración propia.

Artefactos a probar	***Documentación necesaria***
Aplicación Web.	*Especificación de requisitos.* *Especificación de caso de uso del sistema.* *Especificación de requisitos no funcionales.* *Casos de uso del negocio.* *Caso de prueba.*
Manual de usuario.	*Aplicación.*
Manual de instalación.	*Instalador de la Aplicación.*
Glosario de términos.	*Nada.*
Prototipos no funcionales.	*Especificación de CUS.* *Especificación de requisitos.*
Ayuda.	*Manual de usuario.* *Aplicación.*

- Objetivos.

Los objetivos del “Plan de pruebas de aceptación” son:

- Identificar los recursos necesarios para realizar las pruebas de aceptación.
- Identificar los elementos de pruebas y sus prioridades.
- Definir los términos de la aceptación.
- Describir y recomendar las estrategias de las pruebas a ser empleadas.
- Definir el cronograma de las pruebas.

- Alcance.

El alcance que tendrán las pruebas está dado por la intención de ambas partes de revisar únicamente las funcionalidades asociadas a los grupos de casos de usos del sistema. Estas pruebas se proponen realizar en una única etapa, al finalizar las pruebas de unidad, integración y sistema.

- Identificación del proyecto.

El Sistema Unificado de Gestión de Fuerza de Trabajo Calificada está conformado por 5 Módulos. La versión 1.0 de este sistema incluye un gran número de casos de uso, los cuales están recogidos en los documentos de especificación de requisitos de cada uno de los módulos.

- Estrategia de evolución del plan.

El plan será chequeado periódicamente y se realizarán los ajustes que sean pertinentes al cronograma si alguna situación en particular modifica los eventos previstos. Al finalizar cada jornada de trabajo, el equipo de pruebas se reúne para verificar el cumplimiento del plan y realizar dichos ajustes de ser necesarios.

Se realizará un documento con las modificaciones al cronograma, el cual será distribuido entre los participantes de las pruebas para con ello comunicar los cambios.

Las propuestas de modificaciones al plan serán presentadas para su análisis en reunión de chequeo conjunto, participando involucrados del departamento FTC del Ministerio de Economía y Planificación (MEP) y el equipo de desarrollo del proyecto FTC.

- Aprobación de los involucrados.

Tabla # 23: Involucrados en las pruebas de aceptación. Fuente: elaboración propia.

Nombre y Apellidos	Responsabilidad
Carlos Tonet Groero.	Jefe de proyecto.
Virginia Martín Martínez.	Jefa del Departamento de Fuerza de Trabajo Calificada del MEP.

- Roles y responsabilidades.

En la siguiente tabla se muestran los roles, cantidad de personas y responsabilidades del personal que intervendrá en las pruebas.

Tabla # 24: Roles y responsabilidades para pruebas de aceptación. Fuente: elaboración propia.

Rol	Cant.	Responsabilidad
Responsable del proyecto por el ministerio.	1	Controlar el cumplimiento del “Plan de pruebas de aceptación”.
Líder funcional.	1	Avalar la parte funcional del producto en conjunto con los expertos funcionales.
Líder técnico.	1	Monitorear el “Plan de pruebas de aceptación”. Probar la parte técnica del producto. Controlar el escenario de pruebas.

		Emitir propuestas de cambios de manera conjunta con los funcionales.
Especialistas de calidad.	2	Supervisar el trabajo de pruebas, recogiendo las no conformidades para elaborar el informe de conjunto con los funcionales que ejecutarán las mismas. Controlar, monitorear y ejecutar el "Plan de pruebas de aceptación". Diseñar las pruebas del software. Evaluar el proceso de pruebas y los resultados de las mismas.
Expertos funcionales (usuarios finales) pertenecientes al MEP.		Realizar las pruebas al producto. Verificar y validar las funcionalidades de la aplicación. Elaboración de conjunto con los especialistas de calidad el informe final de no conformidades y del informe final de pedidos de cambios.
Equipo de proyecto FTC.		Instruir a todo el equipo de pruebas sobre cómo trabajar con el producto. Recepcionar los informes finales de "No conformidades" y "Pedidos de cambios" aprobados donde quedarán recogidos los elementos detectados y responderlos para ser presentados junto al "Acta de Aceptación" del entregable.

- Estrategia de las pruebas de aceptación.

Para las pruebas de aceptación se realiza una reunión de inicio donde se hará la entrega del producto con todos los entregables a probarse y las descripciones. Durante un tiempo el cliente realiza la revisión y entrega las no conformidades que encuentra. Luego, el equipo de desarrollo debe dar respuesta a las no conformidades, las cuales serán conciliadas en una reunión para ese proceso; terminada dicha reunión solo resta la aceptación del producto que consta en la minuta de la última reunión que se realiza para la aceptación.

- Descripción de las estrategias y tipos de pruebas.

Las pruebas se realizarán de forma manual y se probarán las funcionalidades desarrolladas según la metodología descrita en este documento. Se harán en cada una de las etapas previstas teniendo en cuenta las siguientes fases:

Primera fase: organización del escenario de pruebas y capacitación del equipo de pruebas.
Segunda fase: realización de pruebas de funcionalidad.
El principal objetivo de este tipo de prueba es medir la correspondencia entre el documento "Especificación de CUS" y las funciones que realmente fueron implementadas en el sistema. Como parte de este proceso de verificación se realizarán validaciones sobre la lógica del flujo básico de esta parte del sistema.
Tercera fase: Se realizarán pruebas de seguridad con el objetivo de verificar que sólo los roles de usuarios definidos tengan acceso a sus funcionalidades correspondientes.
Cuarta fase: Conciliación de los documentos entregables de las pruebas, aprobación y firma de los mismos.

Aunque se tiene el "Plan de pruebas de aceptación" aún no se puede proceder a probar, pues para ello el producto debe estar completamente terminado y probado según los anteriores niveles de prueba. De ahí que para la validación de este trabajo de diploma no se cuente con los resultados de la aplicación de las pruebas de aceptación.

3.4. Evaluación de los resultados de las pruebas

Luego de aplicados cada uno de los niveles de prueba se realiza una evaluación de los resultados arrojados para el nivel, donde se recogen de manera resumida los principales defectos encontrados y se declaran fallidas o no las pruebas.

Para las pruebas unitarias se obtuvo el documento "Evaluación de pruebas unitarias", en el que se resume todo lo referente a las deficiencias detectadas y el cual puede ser consultado en Anexo 16 del trabajo.

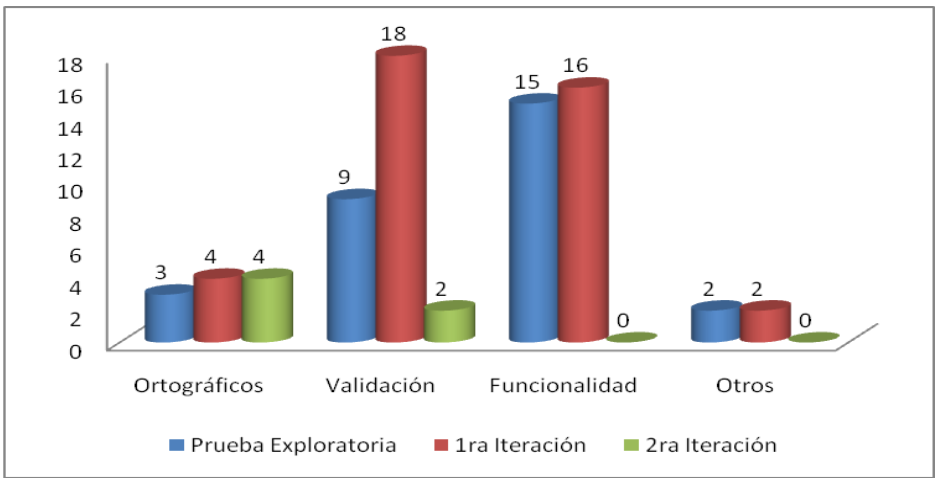

Fig. 7: Evaluación de pruebas unitarias. Fuente: elaboración propia.

La figura anterior muestra gráficamente, de los 64 errores encontrados en las pruebas unitarias, la cantidad que existen de cada tipo, o sea, ortográficos, de validación, de funcionalidad y recomendaciones. Todo esto trajo consigo que las pruebas de unidad se declararan exitosas, debido al gran número de no conformidades encontradas.

En el caso de las pruebas de integración se tiene la evaluación de pruebas de integración que contiene un resumen de los resultados para este nivel y puede ser consultado en el Anexo 17 de este trabajo. Por dichos resultados estas pruebas fueron declaradas exitosas.

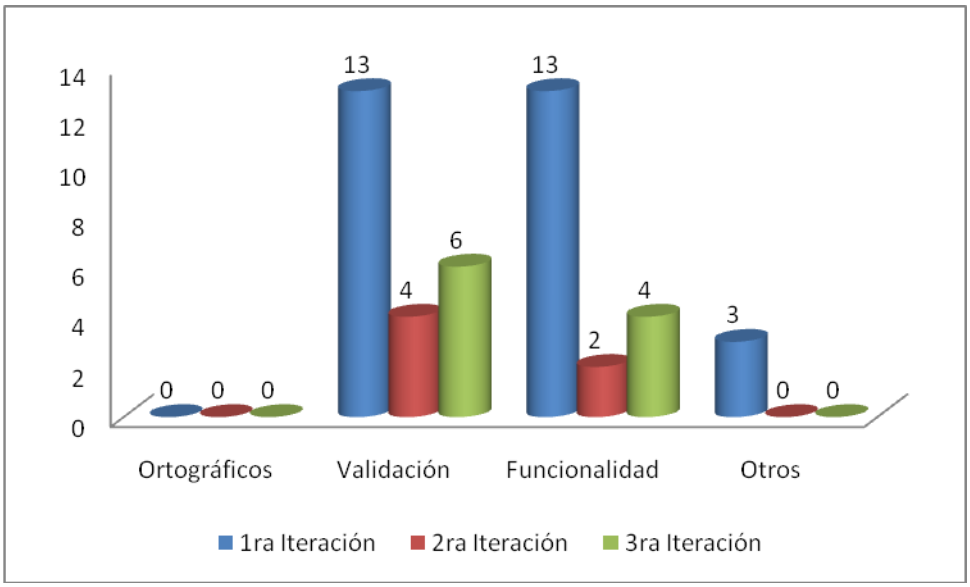

Fig. 9: Evaluación de pruebas de integración. Fuente: elaboración propia.

Luego, al evaluar las pruebas de sistema (ver Anexo 18), se obtuvo que en un ambiente para peticiones http, con el sistema operativo windows, un servidor de base de datos Xeon a 3.0 GHz, 1 GB de RAM y 2 discos duros de 36 y 250 GB, un servidor web Xeon a 3.0 GHz, 1 GB de RAM y 3 discos duros de 36 GB cada uno con RAID 5 y el lenguaje utilizado PHP, se obtuvo que para un total de 125 usuarios (hilos) la aplicación generó un total de 1.81 kb/seg de transferencia de datos lo que incurrió en un rendimiento de 1.4 seg. Se demuestra que la aplicación es estable ya que para estos 125 usuarios se mantuvo prestando servicios todo el tiempo sin incurrir en fallos. Además el tiempo de respuestas del servidor se encuentra por debajo del tiempo que se especificaba en los requisitos del sistema.

El sistema asegura el control de acceso a nivel de usuario, validando que cada usuario al acceder, siempre autenticándose, adquiera los permisos y privilegios que le son permitidos. La arquitectura adoptada posee tratamiento de inyecciones SQL, métodos de encriptamiento, entre otras medidas que

hacen a GeForza un producto de gran seguridad. Además, se obtuvo que el sistema cumple con el 84.2% de los requisitos no funcionales establecidos para los dos módulos de la muestra.

La base de datos mantiene un funcionamiento estable ante la introducción de un millón de datos y aunque se detectaron errores de sintaxis e inserción para un 62% de las tablas existentes, en muchos casos fueron el resultado de que se valida que no se inserten caracteres inválidos.

El nivel restante es el de pruebas de aceptación, el cual no se lleva a cabo debido a que el sistema aún no está listo para presentarse ante la evaluación del cliente. Estas pruebas serán aplicadas guiadas por el "Plan de pruebas de aceptación" diseñado luego que el producto sea liberado.

3.5. Conclusiones

Tras haber aplicado las pruebas propuestas y evaluado los resultados se obtuvo que el producto no da cumplimiento al 15.8% de los requisitos no funcionales, aunque el sistema brinda respuestas a un alto número de peticiones en un tiempo menor al esperado, demostrando la estabilidad del sistema desarrollado. Además, cuenta con la seguridad esperada según lo establecido y con una interfaz amigable y dinámica.

Se comprobó que todos los requisitos funcionales establecidos para los módulos "Administración" y "Sistema matriz" están implementados, aunque persisten funcionalidades deficientes, lo que provoca que el producto aún no esté listo para aplicársele pruebas de liberación y, por tanto, no se puede culminar la aplicación de la estrategia propuesta con las pruebas de aceptación.

Conclusiones

En el presente trabajo se obtuvo una estrategia de pruebas de software para el proyecto FTC con la que se detectaron los defectos existentes en el sistema, dando cumplimiento, así, al objetivo general que era definir e implementar una estrategia de pruebas para detectar los defectos del producto GeForza.

Con la aplicación de la estrategia han sido corregidos, aproximadamente, 80 errores detectados, lo que conlleva que al momento de las pruebas de liberación del producto tenga un número reducido de defectos, lo que corrobora el cumplimiento de la hipótesis planteada.

La estabilidad del sistema ante un alto número de peticiones de usuarios, unido al efectivo control de acceso de usuarios y a la interfaz amigable y de fácil interacción que posee, evidencian el cumplimiento de los requisitos no funcionales establecidos para los dos módulos tomados como muestra.

Se comprobó que la base de datos para todo el producto funciona correctamente para un incremento de 1000000 de datos probados, obteniéndose que los resultados completados correctamente se mantuvieron en el rango del 94% al 100% en cada una de las tablas analizadas.

Se obtuvo que el 92% de las no conformidades encontradas para los dos módulos tomados como muestra sean críticas, de las cuales el 57% son errores de validación y el 36% de funcionalidad.

Se demostró que el 100 % de las funcionalidades establecidas por el usuario para los módulos “Administración” y “Sistema Matriz” fueron implementadas, de las cuales, aún en la tercera iteración de las pruebas de integración, el 38% para el módulo “Sistema Matriz” son deficientes, implicando que, de todo el producto, sólo el módulo “Administración” se encuentre listo para ser sometido a pruebas de liberación.

Recomendaciones

Se recomienda al proyecto FTC aplicar la estrategia propuesta a todos los módulos del producto GeForza incluyendo las pruebas de seguridad con el Shadow Security Scanner y la ejecución de las pruebas de aceptación luego de la liberación del producto.

Se recomienda al Centro de Informatización Universitaria (CENIA) utilizar la propuesta como referencia para cualquier proyecto de gestión que se desee desarrollar, teniendo en cuenta que es sumamente importante aplicar pruebas de caja blanca al código en el momento de la implementación.

Bibliografía referenciada

1. IEEE: Normas y Estándares. Disponible en: (http://calisoft.uci.cu/index.php?view=category&id=12%3Anormas-y-estandares-internacionales&option=com_content&Itemid=24) [En línea: 24/01/10]
2. INGENIERÍA DE SOFTWARE—CALIDAD DEL PRODUCTO—PARTE 1: MODELO DE LA CALIDAD. Disponible en: (http://calisoft.uci.cu/tmp/documentos/normas/iso/NC-ISO-IEC%209126-1.pdf [En línea: 24/01/10]
3. *ISO 8402:* Normativa sobre Calidad, 1986
4. Jacobson, Ivar. Booch, Gady. Rumbaugh, Jame. Proceso unificado de desarrollo de software. Madrid. Addison Wesley ed, 2000.
5. Metodologías de desarrollo de software, Fernández-Medina, Eduardo. Grupo Alarcos. Universidad de Castilla. España, 2007. Disponible en: http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf [En línea:30/09/09]
6. Myers, G *The art of software testing,* 1979.
7. Pressman, Roger S. *autor* Ingeniería del Software: un enfoque práctico. (Español) / La Habana, Editorial McGraw Hill, 2005, ed. 5ª edición.
8. Rojas, Johanna, Barrios, Emilio, *Estándar IEEE de Planes para el Aseguramiento de Calidad del Software", 1984,* Grupo ARQUISOFT, 2007. Disponible en: *(*http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML) [En línea: 10/10/09]
9. Sommerville, Ian. Ingeniería del software, p491-515, 2005. Séptima edición. Disponible en: (http://books.google.com.cu/books?id=gQWd49zSut4C&pg=PA491&dq=pruebas+de+software&lr=#v=onepage&q=pruebas%20de%20software&f=false) [En línea:30/09/09]

Bibliografía consultada

1. Amo, Alonso, Martínez Norman, Loic. Introducción a la Ingeniería del software, pagina 87, 2005. Disponible en: http://books.google.com.cu/books?id=rXU-WS4UatYC&pg=PA87&dq=pruebas+de+software&cd=3#v=onepage&q=pruebas%20de%20software&f=false [En línea: 10/12/09]
2. Artola, Luis. Tipos de pruebas automatizadas de software, 2009. Disponible en: http://www.programania.net/diseno-de-software/tipos-de-pruebas-automatizadas-de-software/ . [En línea: 12/01/10]
3. Black, Rex. Critical Testing processes. Addison-Wesley, 2004.
4. Conceptos generales de calidad total. Disponible en: http://www.monografias.com/trabajos11/conge/conge.shtml [En línea: 8/12/09]
5. Dolado Cosín, Javier, Tuya González, Javier, Ramos, Isabel. Técnicas cuantitativas para la gestión en la Ingeniería de software, Ed. Cristina seco López, España, 2007. Disponible en: http://books.google.com.cu/books?id=PZQoZ9KTNaEC&printsec=frontcover&dq=inauthor:%22Javier+Dolado+Cos%C3%ADn%22&cd=1#v=onepage&q&f=false [En línea:30/09/09]
6. Fernández Carrasco, Oscar M., García León, Delba y Beltrán Benavides, Alfa. Un enfoque actual sobre la calidad del software. Disponible en: http://www.bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.html [En línea: 10/12/09]
7. Gómez Gallego, Juan Pablo. Fundamentos de la metodología RUP. Universidad Tecnológica de Pereira, 2007. Disponible en: http://www.scribd.com/doc/297224/RUP [En línea: 12/01/10]
8. Guzmán Cortés, Oscar Hernando. Aplicación práctica del diseño de pruebas de software a nivel de programación, 2004. Disponible en: http://www.willydev.net/descargas/oguzman-diseno_pruebas.pdf. [En línea: 10/10/09]
9. Guzmán, Martín, Tojin, Karen A., Hurtarte, Héctor, Sánchez, Kevin. Metodologías para análisis y diseños orientados a objetos y MDA. Universidad del Valle de Guatemala, 2009. Disponible en: http://www.scribd.com/doc/12848359/Metodologias-Para-Analisis-y-Diseno-Orientado-a-Objetos-y-MDA. [En línea:30/09/09]
10. Grimán, Anna C., Pérez, María, Mendoza, Luis E. Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales, Laboratorio de Investigación de Sistemas de Información (LISI), Departamento de Procesos y Sistemas, Universidad Simón Bolívar, Venezuela. http://www.lisi.usb.ve/publicaciones/03%20evaluacion/evaluacion_15.pdf [En línea: 30/11/09]

11. IEEE Standard Glossary of Software Engineering Terminology. Disponible en: http://calisoft.uci.cu/tmp/documentos/normas/ieee/610-12-1990_Standard_Glosary_of_SET.pdf [En línea: 24/01/10]
12. ISO 9001:2008 - Sistemas de Gestión de la Calidad – Requisitos. Disponible en: http://www.normas9000.com/iso-9001-2008.html [En línea: 10/10/2009]
13. ISO 9004:2000 - Sistemas de Gestión de la Calidad - Guía de mejoras del funcionamiento
14. Mendoza, Luis E., Grimán, Anna C., Pérez, María. Algoritmo para la Evaluación de la Calidad Sistémica del Software, Laboratorio de Investigación de Sistemas de Información (LISI), Departamento de Procesos y Sistemas, Universidad Simón Bolívar, Venezuela. Disponible en: http://www.lisi.usb.ve/publicaciones/02%20calidad%20sistemica/calidad_21.pdf [En línea: 30/11/09]
15. Polo Usaola, Macario, Mantenimiento Avanzado de Sistemas de Información: Pruebas del Software, Departamento de Informática, Paseo de la Universidad, Ciudad Real. Disponible en: http://alarcos.inf-cr.uclm.es/doc/masi/doc/lec/parte5/polo-apuntesp5.pdf. [En línea: 30/11/09]
16. Raja Prado, Elena. Casi todas las pruebas del software, Actas de Talleres de Ingeniería del Software y Bases de Datos, Vol. 1, No. 4, 2007. Disponible en: http://www.sistedes.es/TJISBD/Vol-1/No-4/articles/pris-07-raja-ctps.pdf. [En línea: 10/10/09]
17. Rojas, Johanna, Barrios, Emilio. Métodos de pruebas de caja negra, Grupo ARQUISOFT, 2007. Disponible en: http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html [En línea: 10/01/10]
18. Sánchez garreta, José Salvador. Ingeniería de proyectos prácticos: actividades y procedimientos, 5ta edición, 2003. Disponible en: http://books.google.com.cu/books?id=MXTI43ThoS4C&pg=PA68&dq=ingenieria+de+software+un+enfoque+practico#v=onepage&q=&f=false [En línea:30/09/09]
19. Selenium IDE: plugin para Firefox para realizar testing. Disponible en: (http://sentidoweb.com/2008/05/19/selenium-ide-plugin-para-firefox-para-realizar-testing.php) [En línea: 12/04/10]
20. Software Lifecycle Optimization. Process Perspective, 2010. Disponible en: http://www.als-es.com/home.php?location=servicios/procesos/gestion-pruebas. [En línea: 12/01/10]
21. The network vulnerability scanner. Disponible en:

http://translate.google.com.cu/translate?hl=es&sl=en&u=http://www.nessus.org/&ei=NCfDS7K5AcaAlAftoozbBA&sa=X&oi=translate&ct=result&resnum=1&ved=0CA8Q7gEwAA&prev=/search%3Fq%3DNessus%26hl%3Des En línea: 12/04/10]

Glosario de términos

- Artefacto: Documento que registra los resultados de la prueba.
- ASD: Desarrollo de Software Adaptativo.
- CU: Caso de uso.
- CUS: Caso de Uso del Sistema.
- DCPCE: Diseño de casos de prueba de carga y stress.
- DCPS: Diseño de casos de prueba de seguridad.
- DSDM: Método de Desarrollo de Sistemas Dinámicos.
- Feature Driven Development (FDD): metodología ágil para el desarrollo de sistemas de software.
- FTC: Fuerza de Trabajo Calificada.
- GeForza: Sistema Unificado de Gestión de Fuerza de Trabajo Calificada.
- HTML**:** Lenguaje de Marcado de Hipertexto.
- Lean Development (LD): metodología ágil para el desarrollo de sistemas de software.
- MSF: Marco de Trabajo para Software de Microsoft .
- No conformidad: defecto detectado que afecta la calidad del producto.
- RUP: Proceso Unificado de Desarrollo.
- SCRUM (Wicked Problems Righteous Solutions): metodología de desarrollo que propone soluciones fáciles a problemas difíciles.
- XP: Programación Extrema.
- Pruebas de liberación: pruebas que se le realizan al producto para comprobar que cumple con los requisitos establecidos y que está listo para ser entregado al cliente.