Universidad de las Ciencias Informáticas

Facultad 7

Diseño, ejecución y evaluación de casos de prueba del Registro de Población de la Atención Primaria del Sistema de Información para la Salud.

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

Autoras: Darlen Hornia Puentes
Yisel Jorge Basulto
Tutores: Lic. Lucy Cruz Águila
Ing. David Barreto Medina
Asesor: Yoenny Pérez

Ciudad de La Habana, Julio del 2007

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los 5 días del mes de julio del año 2007.

Autores

_____	_____
Darlen Hornia Puentes	Yisel Jorge Basulto

Tutores

_____	_____
Lic. Lucy Cruz Águila	Ing. David Barreto Medina

DATOS DE CONTACTO

Tutor: Lic. Lucy Cruz Águila

Especialista de calidad de la dirección de Desarrollo de la Empresa SOFTEL, graduada de Licenciatura en Cibernética-Matemática en el año 1988. Ha desarrollado diferentes proyectos de gestión en la esfera de la salud, turismo, empresarial, ministerial, ha dirigido técnicamente proyectos de gestión en estas áreas. Fue directora de tecnología y calidad de la empresa SOFTEL en el período del 2000 al 2003. Ha participado en diferentes eventos científicos-técnicos en temas de calidad del desarrollo de software. Ha recibido diferentes cursos de post-grados en Ingeniería de Software, programación y calidad.
Correo electrónico: lucycruz@softel.cu
Ubicación: SOFTEL, UCI, Cuba

Tutor: Ing. David Barreto Medina

Ingeniero Industrial graduado en la UHO “Oscar Lucero Moya”, en el año 2004. Posee 3 años de experiencia laboral, se ha desempeñado como profesor en la Universidad de las Ciencias Informáticas, UCI. Durante este mismo tiempo ha estado desarrollando conjuntamente con un equipo de trabajo de la UCI y de la empresa SOFTEL una solución informática que automatice y gestione los procesos inherentes a la Atención Primaria de la Salud en Cuba, donde ha fungido como Líder de Proyecto.
Correo electrónico: dbarreto@uci.cu
Ubicación: Facultad 7, UCI, Cuba

Asesor: Ing. Yoenny Perez Romero:

Ingeniero Informático graduado en el Instituto Superior Politécnico “José Antonio Echeverría”, en el año 2005. Posee 2 años de experiencia laboral, se ha desempeñado como profesor en la Universidad de las Ciencias Informáticas, UCI, vinculado siempre a las asignaturas de la especialidad. Durante este mismo tiempo ha estado desarrollando conjuntamente con un equipo de trabajo de la UCI y de la empresa SOFTEL una solución informática que automatice y gestione los procesos inherentes a la Atención Primaria de la Salud en Cuba, donde ha fungido como Líder de Proyecto.
Correo electrónico: yoenny@uci.cu
Ubicación: Facultad 7, UCI, Cuba

"El hombre cauto jamás deplora el mal presente; emplea el presente en prevenir las aflicciones futuras".

William Shakespeare

DEDICATORIA

A mis padres Malena y Rubinelson por su amor incondicional, su infinita ternura y por confiar en mí hasta el último momento.

A mi hermano Yander y mi novio Andy por el apoyo y comprensión.

A mis abuelitas Zoila y Carmelina y mi abuelito Rubén por sus valiosos consejos.

Yisel Jorge Basulto

A mis tíos Adria y Frank por estar a mi lado en todo momento, por apoyarme y ayudarme con sus consejos siempre que me hizo falta.

A mis padres Marlene y Degni y a mis abuelos Nena y Kiki por confiar en mí y tener paciencia para poder comprenderme.

A mis amigas Karelys, Yilén, Lucy y Yaimí por acompañarme siempre y darme ánimos para seguir adelante.

Darlen Hornia Puentes

AGRADECIMIENTOS

A nuestra tutora por la confianza depositada en nosotras.

A nuestros compañeros Yoiler y Johander por su ayuda incondicional.

A nuestras amigas por acompañarnos en los buenos y malos momentos.

En fin a todas aquellas personas que de una forma u otra han colaborado con la realización de este trabajo.

RESUMEN

El presente trabajo de diploma tiene como objetivo ejecutar el proceso de prueba del Registro de Población de la Atención Primaria del Sistema de Información para la Salud, específicamente a la opción Historia de Salud Familiar donde se gestionan todos los datos de los pacientes y sus Historias de Salud Familiar. Se ejecutan cada una de las actividades del Flujo de trabajo de prueba generándose los artefactos resultantes de cada actividad.

El proceso se realizó mediante casos de prueba de caja blanca y caja negra, diseñados utilizando las técnicas del Camino Básico y Partición Equivalente respectivamente, estos casos constituyeron la entrada para su ejecución a través de un componente de prueba implementado y la interfaz correspondiente al módulo que se prueba. Se archivaron los resultados obtenidos de la ejecución del proceso y se realizó una evaluación de cada parte probada. La estrategia y los recursos necesarios para llevar a cabo este proceso se planificó en el Plan de prueba, estableciendo una base para el control y seguimiento durante la ejecución.

Este proceso garantizará un producto de software competitivo, con la calidad requerida y con un alto nivel técnico. Motivará un proceso de Gestión de Cambio paralelo al proceso de prueba, mediante el cual se irán mitigando los defectos resultantes del proceso de desarrollo. Además servirá de guía para extender este proceso a todas las opciones del Registro de Población del Sistema de Información para la Salud.

Palabras claves: **Casos de prueba**, **Plan de prueba y Componente de prueba**.

ÍNDICE GENERAL

INTRODUCCIÓN

La Informatización de la sociedad es de vital importancia y prioridad para el estado cubano. Para llevarla a cabo es necesario asumir las exigencias que el desarrollo tecnológico actual impone. Además, se debe capacitar al personal, recibir y explotar las nuevas tecnologías de punta que se crean, aprovechando al máximo los recursos tecnológicos puestos a la disposición de la sociedad, por la Revolución, así como la ampliación de las redes informáticas y la creación de nuevas redes para los diferentes sectores de la sociedad.

La Dra. Alina Ruiz Jhones, Vicerrectora Primera de la Universidad de Ciencias Informáticas, en la inauguración del III Taller de Calidad de las Tecnologías de la Información y las Comunicaciones en el marco de la XII Convención y Exposición Internacional, INFORMÁTICA 2007, hizo referencia a las siguientes ideas transmitidas por el Comandante en Jefe Fidel Castro Ruz en su visita a la Universidad de las Ciencia Informáticas (UCI):

(…)"Las producciones intelectuales serán el sustento fundamental de Cuba. La idea es convertir la informática en una de las ramas más productivas y portadoras de recursos para la nación". (…)

(…)"Estos son los profesionales más importantes que vamos a formar en cuanto a perspectiva económica". (…)

(…)"Nuestra sociedad será una sociedad de trabajadores intelectuales. Dando vueltas y meditando - no es mucho lo que todavía hemos pensado y profundizado -, parece ser que aquí está la cantera principal". (…)

(…)"La UCI, sería mejor decir la Informática, se convertirá en una poderosísima fuerza científica, económica e incluso política, para el país". (…)

La Universidad se crea en el año 2002, con las siguientes misiones:

-Formar profesionales, comprometidos con su Patria, altamente calificados en la rama de la informática.

-Producir software y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación.

Introducción

A diario se esfuerzan por hacer realidad las palabras del Comandante en Jefe con el objetivo de construir un futuro mejor para la sociedad en que vive. En la Universidad de las Ciencias Informáticas se llevan a cabo varios proyectos de producción de software para las diferentes ramas de la economía, que contribuirán en gran medida a la informatización de la sociedad y aportarán grandes beneficios a la economía del país. Es por ello que el deber fundamental es cumplir estas misiones teniendo bien presente la importancia y trascendencia de la tarea que la Revolución ha puesto en las manos de los estudiantes que aquí se preparan.

Como parte de la Informatización de la sociedad está la Informatización de la Salud Pública. El perfeccionamiento de los sistemas informativos y la informatización de la salud está bajo el asesoramiento del Ministerio de la Informática y las Comunicaciones (MIC) que propone, con dominio de la actualidad en la ingeniería de software, la metodología y las aplicaciones más convenientes para el desarrollo de esta esfera en el sector de la Salud con las buenas prácticas en la producción de software con alta calidad, todo esto en estrecho vínculo con médicos o especialistas de la salud con conocimiento del ramo para que se lleven a cabo las acciones pertinentes a la hora de captar requerimientos, proponer modelos y programar. [1]

Dentro de la Informatización del sector de la Salud Pública Cubana, se lleva a cabo la realización de un Sistema de Información para la Salud (SISalud) que aportará beneficios tales como: aumentar la calidad de vida de la población, la accesibilidad y la equidad de los servicios, aumentar los conocimientos científicos y la efectividad de los profesionales, permitiendo de esta manera al Ministerio de Salud Pública tomar decisiones en los diferentes niveles con la mayor eficacia y rapidez posible ante cualquier situación que se presente, así como mejorar los servicios médicos que se brindan a la población.

El Registro de Población (RPOB) es unos de los módulos en desarrollo que forma parte de este sistema. La importancia de su surgimiento viene dada porque le permitirá al Equipo Básico de Salud (EBS), binomio conformado por el médico y la enfermera de la familia, atender a los pacientes de una forma más rápida y eficiente evitando todo el "papeleo" que actualmente existe y que se deteriora con el paso del

[1] Derivet, D., Cabrera, M. y Marin, M. Atención Primaria de la Salud. 2007

tiempo. Este Registro posibilita la dispensarización (Clasificación de los pacientes que están registrados en la Historia de Salud Familiar, HSF, de acuerdo a la enfermedad, en grupos dispensariales) de los pacientes, siendo este el principal proceso que gestiona, o sea cada paciente es ubicado en el grupo dispensarial al cual pertenece siendo de mucha importancia pues así se saben los problemas de salud que tienen y se le puede dar un seguimiento a los mismos. Además le permite al Departamento de Estadística hacer reportes estadísticos, los cuales son muy importantes para las investigaciones y los análisis que en términos de salud se hacen.

El Módulo Población gestiona la información de la Historia de Salud Familiar de los Equipos Básicos de Salud permitiendo la captación, y organización de datos relevantes para conocer el estado de salud de cada familia y de la comunidad.

Este módulo beneficiará al paciente y las familias por cuanto:

- Se aplicarán eficientemente los programas de salud existentes, las acciones de salud serán mejor planificadas de acuerdo a las situaciones reales, modificables o no, de la población, facilitando un mejor seguimiento de los pacientes, logrando el objetivo de mejorar la atención y por tanto, elevar la calidad de vida y de los servicios que se brindan.
- Se dispondrá de la información referente a antecedentes patológicos tanto individuales como familiares de tipo biológicos, psicológicos, ambientales y sociales lo cual proporciona una base más sólida a los médicos para entender mejor al paciente y su proceso salud-enfermedad para tomar en consecuencia las decisiones más apropiadas en cada caso.

En el país se están realizando inversiones para convertir la producción de software en un renglón importante de la economía por lo que se hace cada vez mayor la necesidad de obtener productos de software que sean desarrollados con profesionalidad, que tengan un alto nivel de calidad, productos realmente competentes. Esto se puede lograr llevando a cabo el proceso de desarrollo de software haciendo uso de las mejores prácticas desarrolladas profesionalmente.

El desarrollo de sistemas de software implica una serie de actividades de producción en las que las posibilidades de que aparezca el fallo humano son enormes. Los errores pueden empezar a darse desde

el primer momento del proceso, en el que los objetivos pueden estar especificados de forma errónea o imperfecta, así como en posteriores pasos de diseño y desarrollo. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo de software ha de ir acompañado de una actividad que garantice la calidad. Las pruebas del 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. [2]

La creciente percepción del software como un elemento del sistema y la importancia de los costos asociados a un fallo del propio sistema, están motivando la creación de pruebas minuciosas y bien planificadas.

Las revisiones, descubren errores, pero no son suficientes. Cada vez que el programa se ejecuta, el cliente lo está probando. Por lo tanto, se debe hacer un intento especial por encontrar y corregir todos los errores antes de entregar el sistema al cliente. Con el objetivo de encontrar el mayor número posible de errores, las pruebas deben conducirse sistemáticamente, y los casos de prueba deben diseñarse utilizando técnicas definidas.

Es importante profundizar en el conocimiento de las técnicas de pruebas del software. Estas técnicas facilitan una guía sistemática para diseñar pruebas que: comprueben la lógica interna de los componentes software y verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y rendimiento.

Un proceso de pruebas requiere mucho más que tiempo y dinero, necesita una verdadera metodología la cual exige herramientas y conocimientos destinados a dicha tarea.

Teniendo en cuenta que en el proyecto SISalud hay un conjunto de módulos que están en fase de liberación, en los cuales el proceso de prueba no se hace de manera organizada, y se omiten las buenas

[2] Pressman, R. S. Ingeniería del Software. Un enfoque práctico. La Habana, Cuba 2005.

prácticas que exige este flujo de trabajo se toma la decisión de llevar a cabo el Flujo de trabajo de prueba en el módulo RPOB (Registro de Población) por la importancia que merita toda la información que aquí se gestiona relacionada con el paciente para garantizar un producto de software competitivo, con la calidad requerida y con un alto nivel técnico.

Problema: Ejecutar el proceso de prueba aplicando las buenas prácticas al módulo Registro de Población del Sistema de Información para la Salud.

Objeto de estudio: Proceso de pruebas para el Sistema de Información para la Salud.

Campo de acción: Proceso de pruebas para la opción Historia de Salud Familiar del módulo Registro de Población de la Atención Primaria del Sistema de Información para la Salud.

Objetivo General:

Ejecutar el proceso de prueba del Registro de Población de la Atención Primaria del Sistema de Información para la Salud que garantizará un producto con la calidad requerida y un alto nivel técnico.

Para dar cumplimiento a este objetivo se plantean las siguientes tareas:

1. Analizar las buenas prácticas en la ejecución del flujo de trabajo de prueba.

2. Analizar las actividades personales relacionadas con las pruebas definidas en el proceso de software personal (PSP)

3. Analizar las actividades de equipo relacionadas con las pruebas definidas en el proceso de software en equipo (TSP)

4. Vincular el flujo de trabajo de pruebas con PSP y TSP.

5. Analizar las estrategias, tipos de pruebas, niveles de pruebas.

6. Diseñar los casos de prueba de la opción Historia de Salud Familiar (HSF) del módulo Registro de Población de la Atención Primaria del Sistema de Información para la Salud.

7. Ejecutar y evaluar los casos de prueba diseñados.

Resultados esperados:

a. Resultados de las pruebas realizadas a la opción Historia de Salud Familiar (HSF) del Registro de Población de la Atención Primaria del Sistema de Información para la Salud.
b. Evaluación de los resultados.
c. Evaluación del flujo de prueba implementado.

El contenido de este trabajo se encuentra estructurado de la siguiente manera:

En el capítulo 1, “Fundamentos Teóricos” se realiza un estudio de todos los aspectos teóricos relacionados con el Flujo de trabajo de prueba y se abordan los conceptos asociados al proceso de Prueba.

En el capítulo 2, “Plan de prueba” se define el Plan de prueba para la primera iteración de la fase de Construcción de la opción HSF del módulo RPOB en el cual se traza la estrategia para llevar a cabo las principales actividades de este Flujo de trabajo.

En el capítulo 3, “Diseño, ejecución y evaluación de las pruebas” se diseñan y ejecutan los casos de prueba para la opción HSF del módulo RPOB y finalmente se realiza una evaluación de los resultados obtenidos durante la ejecución de los casos de prueba diseñados.

CAPÍTULO
FUNDAMENTOS TEÓRICOS

Este capítulo describe las buenas prácticas en la ejecución del flujo de trabajo de prueba definiendo los roles, actividades y artefactos. Además se ofrece una visión de la estrategia a seguir para ejecutar el proceso de prueba, los tipos de prueba que existen así como los métodos que se aplican en cada tipo de prueba.

1.1 Qué son las pruebas

Las pruebas constituyen una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.[3]

Una buena prueba según Kaner, Falk y Nguyen posee los siguientes atributos:

1. Tiene una alta probabilidad de encontrar un error.
2. No debe ser redundante.
3. Debería ser "la mejor de la cosecha".
4. No debería ser ni demasiado sencilla ni demasiado compleja.

1.2 Objetivos de las pruebas

Pessman para definir los objetivos de las pruebas hace referencia a Glem Myers quien establece varias normas que pueden servir adecuadamente como objetivos de la prueba:

[3] Igual a Referencia 2

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

La prueba no puede asegurar la ausencia de defectos; solo pueden demostrar que existen defectos en el software.

1.3 Comportamiento de la pruebas

1.3.1 En el flujo de trabajo de pruebas desde el punto de vista dinámico

Los ingenieros de pruebas comienzan planificando el esfuerzo de prueba en cada iteración y describen luego los casos de pruebas necesarios y sus procedimientos de pruebas. Si es posible se crean a continuación los componentes de pruebas para automatizar algunos de los procedimientos de prueba. Todo esto se hace para cada construcción entregada como resultado del flujo de trabajo de implementación.

Con estos casos, procedimientos y componentes de prueba como entrada, se prueba cada construcción y se detecta cualquier defecto. Los defectos se usan como retroalimentación tanto para otros flujos de trabajo (diseño, implementación) como para el flujo de pruebas, para que se lleve a cabo una evaluación sistemática de los resultados de las pruebas.

1.3.2 En la Fase de Inicio

El grupo de prueba se va poniendo al corriente de la naturaleza general del sistema propuesto, va considerando qué pruebas requerirá y va desarrollando algunos planes provisionales de prueba. No se realiza en esta etapa un trabajo significativo de pruebas ya que el prototipo exploratorio de demostración tiene por lo general carácter ilustrativo más que operativo.

Se puede generar un modelo de pruebas algo rudimentario en esta fase.

1.3.3 En la Fase de Elaboración

El objetivo es asegurarse de que los subsistemas de todos los niveles y de todas las capas (desde la capa del sistema hasta las capas específicas de la aplicación) funcionen. Sólo se pueden probar los componentes ejecutables.

Al empezar por las capas más bajas de la arquitectura se prueban los mecanismos de distribución, almacenamiento, recuperación y concurrencias de objetos, así como otros mecanismos de las capas inferiores del sistema. Con esto no solo se prueba la funcionalidad sino también el rendimiento. Todas las capas no son necesarias de probar sino es necesario probar cómo las capas superiores hacen uso de las inferiores.

Al planificar las pruebas se seleccionan los objetivos que evaluarán la línea base de la arquitectura.

Al diseñar las pruebas se toman como base estos objetivos para identificar los casos de pruebas necesarios y preparar procedimientos de pruebas para comprobar la sucesiva integración de subsistemas hasta completar la línea base.

Al comprobar los componentes se quedará listo para realizar las pruebas de integración**.**

Al ser integrado el sistema tal y como queda definido por los casos de uso arquitectónicamente significativos se realizan las pruebas de sistema.

1.3.4 En la Fase de Construcción

Al planificar las pruebas se seleccionan los objetivos que comprueben las sucesivas construcciones, y por último el propio sistema.

Al diseñar las pruebas se determina cómo probar los requisitos en el conjunto de construcciones. Se preparan casos y procedimientos de pruebas con este fin.

Se realizan las pruebas de integración informando los resultados para tomar las medidas necesarias en casos de errores.

Se realizan también pruebas del sistema al alcanzarse el status de versión parcial del sistema, informando los resultados para tomar las medidas necesarias en casos de errores.

En esta fase se deben evaluar las pruebas a medida que transcurren las pruebas de integración y de sistema comprobando que éstas alcancen los objetivos del plan de pruebas. Si una prueba no alcanza sus objetivos, los casos y procedimientos de pruebas deberán ser modificados para lograrlos.

1.3.5 En la Fase de Transición

Se pueden realizar pruebas Alfa (se realizan en la empresa que desarrolla el software con la participación del cliente en un ambiente controlado) y/o pruebas Beta (prueba realizadas en organizaciones representativas “clientes beta”) y/o validaciones por terceros (una empresa especializada en pruebas realiza pruebas de aceptación por encargo del cliente). Se recopilan y analizan los resultados de estas pruebas con el objetivo de llevar a cabo acciones.

En esta fase se buscan pequeñas deficiencias que pasaron desapercibidas durante la fase de construcción y que pueden ser corregidas en el marco de la línea base de la arquitectura existente.

1.4 Roles, actividades y artefactos

1.4.1 Roles

Cada proyecto en desarrollo debe crear un grupo de prueba con roles bien definidos, independientemente de que las pruebas pueden ser realizadas por usuarios finales y otros trabajadores del equipo de desarrollo.

1.4.1.1 Administrador de Prueba:

Es el responsable de dirigir técnicamente al grupo de prueba y administrar el proceso. Participa en el proceso de planificación de las pruebas siendo el encargado de elaboración del plan, participa en el proceso de evaluación de las pruebas, siendo el responsable de documentar este proceso. Se encarga de adquirir los recursos apropiados para ejecutar el proceso. Identifica los elementos que motiven o faciliten

las pruebas. Evalúa y audita la calidad del proceso proponiendo mejoras para próximas iteraciones, mantiene informado al jefe de proyecto del estado del proceso de prueba.

1.4.1.2 Analista de Prueba:

Es responsable de identificar los objetivos de las pruebas así como las ideas de las pruebas. Identifica y define las pruebas requeridas en la iteración. Participa en las actividades de planificación, diseño y evaluación de las pruebas.

1.4.1.3 Diseñador de prueba:

Es responsable de la integridad del modelo de pruebas asegurando que el modelo cumple con su propósito. Del modelo de pruebas define y describe los casos de prueba y los procedimientos de prueba. Realiza la gestión y mantenimiento del entorno de los datos (base de datos) de prueba. Administra la base de datos de prueba. Responsable de la elaboración de los componentes de prueba para los procedimientos que puedan ser automatizados.

Participa en las actividades de planificación, diseño, implementación y evaluación de las pruebas.

1.4.1.4 Probador:

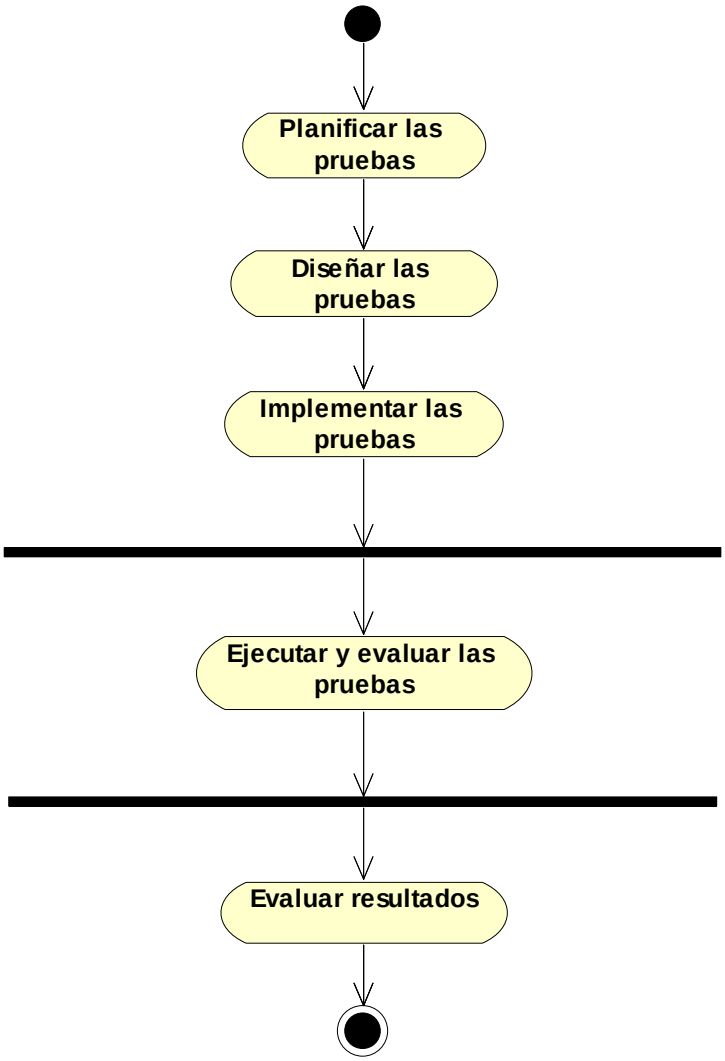
Es el responsable de ejecutar los casos de prueba guiados por los procedimientos de pruebas de forma manual o automática haciendo uso de los componentes de pruebas obtenidos como resultado de la implementación de las pruebas o utilizando herramientas de pruebas identificadas. Participa en la actividad de ejecución de las pruebas siendo responsable de identificar y listar los errores encontrados.

1.4.2 Actividades

De forma general se proponen cinco grandes pasos para realizar las pruebas:

1. Planificar las pruebas de cada iteración obteniendo un plan de prueba por iteración.
2. Diseñar las pruebas con la elaboración de casos de prueba, especificar cómo realizar las pruebas.
3. Evaluar la utilización de algún software que permita automatizar las pruebas o construir uno.

4. Realizar las pruebas y documentar los resultados.
5. Evaluar los resultados de las pruebas realizadas.



.

Figura 1.1 Flujo de trabajo de prueba.

1.4.2.1 Planificar pruebas

Al planificar las pruebas, se planifican los esfuerzos de prueba en una iteración y deben llevarse a cabo las siguientes tareas:

- Describir una estrategia de prueba de la iteración definiendo el método apropiado para diseñar y ejecutar las pruebas, los objetivos, los tipos de pruebas posibles a realizar así como los niveles y el enfoque de las pruebas y las pautas a seguir.
- Estimar los requisitos para el esfuerzo de la prueba, ejemplo recursos humanos, la infraestructura de prueba tanto de hardware como software.
- Definir cronograma de trabajo
- Definir las responsabilidades de cada miembro del grupo de prueba.

Para obtener el plan de prueba se debe partir de los artefactos de entrada: requisitos funcionales, requisitos no funcionales o adicionales, documento visión, modelo de casos de uso, modelo de análisis, de diseño, de implementación, descripción de la arquitectura y otros artefactos o documentos disponibles que puedan aportar al proceso de planificación de las pruebas.

Como artefacto de salida de esta actividad se obtiene el plan de prueba.

Los miembros del grupo de prueba que deben participar en esta actividad son los que desempeñan los roles: administrador de prueba, analista de prueba y diseñador de prueba.

1.4.2.1.1 Estrategia de Prueba

La estrategia de prueba describe el enfoque y los objetivos generales de las actividades de prueba**.** Incluye los niveles de prueba (unidad, integración, sistema, aceptación) a ser diseccionados y el tipo de prueba a ser ejecutada.

La estrategia define:

- Técnicas de pruebas (manual o automática) y herramientas a ser usadas.
- Qué criterios de éxitos y culminación de la prueba serán usados.

- Consideraciones especiales afectadas por requerimientos de recursos o que tengan implicaciones en la planificación.[4]

El enfoque de los diferentes tipos de pruebas se realiza en dependencia del número de iteraciones, el tamaño de la iteración y el tipo de proyecto que se está probando.

Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta ejecución del proceso de pruebas.

Cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de pruebas y la depuración y evaluación de los datos resultantes.

Una estrategia de prueba propone movernos hacía afuera en una espiral, de manera que primero se prueban las unidades más pequeñas del diseño del software y después cómo se integran los componentes en los cuales están contenidas estas unidades.

1.4.2.2. Diseñar pruebas

Al diseñar las pruebas deben tenerse los siguientes propósitos:

I. Identificar los casos de prueba para cada construcción

Para identificar los casos de prueba de cada construcción los se debe partir de los artefactos de entrada: requisitos adicionales, modelo de casos de uso, modelo de análisis, modelo de diseño, modelo de implementación, descripción de la arquitectura (vistas arquitectónicas de los modelos) y del plan de pruebas. De esa forma deben obtener los casos de pruebas.

[4] Colectivo de Profesores. Culminación de la Fase de Elaboración
Flujo de trabajo de Prueba. 2005

II. Identificar y estructurar los procedimientos de prueba especificando cómo realizar los casos de prueba

Se debe reutilizar procedimientos de prueba existentes tanto como sea posible, realizando las modificaciones adecuadas. Se deben crear procedimientos de pruebas que puedan ser reutilizados en varios casos de prueba.

Los procedimientos de prueba deben ser descritos definiendo los objetivos del caso de prueba, los datos de entrada, condiciones de ejecución, los resultados esperados. Las técnicas y tipos de prueba que se ejecutarán para ejecutar el caso de prueba.

Los miembros del grupo de prueba que deben participar en esta actividad son los que desempeñan los roles de diseñador de prueba y analista de prueba.

1.4.2.3. Implementar pruebas

El propósito de la implementación de pruebas es automatizar uno o varios procedimientos de prueba, creando componentes de pruebas cuando sea posible. Lo que ahorraría tiempo y esfuerzo a la hora de ejecutar las pruebas.

Los componentes de pruebas usan por lo general grandes cantidades de datos de entrada y generan grandes cantidades de datos de salida para visualizar estos datos en ocasiones es útil utilizar hojas de cálculo y bases de datos. Estos componentes son muy útiles a la hora de realizar las pruebas de regresión.

Como artefacto de entrada a esta actividad se tienen los procedimientos de prueba y como artefacto de salida se obtienen los componentes de prueba.

Los miembros del equipo de prueba que deben participar en esta actividad son los que desempeñan los roles de analista de prueba y diseñador de prueba.

1.4.2.4. Ejecutar pruebas

Esta actividad garantiza que se realicen las pruebas diseñadas para cada una de las construcciones creadas en la iteración y se recopilan los resultados de las pruebas. Estas pruebas se llevan a cabo con los siguientes pasos:

1. Realizar las pruebas realizando los procedimientos de pruebas manualmente o ejecutando los componentes de pruebas si existen.
2. Comparar los resultados de las pruebas con los esperados e investigar los resultados de las pruebas que no coinciden con los esperados.
3. Documentar los defectos detectados.

Los artefactos de entrada de esta actividad son los casos de prueba, los procedimientos de prueba, los componentes de prueba y el plan de prueba; y como artefacto de salida la lista de defectos.

1.4.2.5. Evaluar prueba

El propósito de esta actividad es evaluar los esfuerzos de prueba en una iteración. Para evaluar los resultados de pruebas estos deben ser comparados con los objetivos esbozados en el plan de pruebas. Los diseñadores de pruebas preparan métricas que les permiten determinar el nivel de calidad de software y qué cantidad de pruebas es necesario hacer.

El administrador de prueba observa dos métricas:

1. Compleción de la prueba: obtenida a partir de la cobertura de los casos de prueba y de la cobertura de los componentes probados. Esta métrica indica el porcentaje de casos de prueba que han sido ejecutados y el porcentaje de código que ha sido probado.
2. Fiabilidad: la cual se basa en el análisis de las tendencias en los defectos detectados y en las tendencias en las pruebas que se ejecutan con el resultado esperado.

Los artefactos de entrada a esta actividad son el plan de prueba, la lista de defectos y los procedimientos de prueba, como artefacto de salida se tiene la evaluación de los resultados de las pruebas de la iteración.

En esta actividad participa el grupo de prueba siendo responsabilidad del administrador documentar la evaluación.

1.4.3 Artefactos

Como resultado de la realización de las actividades incluidas en este flujo de trabajo, los trabajadores construyen los siguientes artefactos:

1.4.3.1 Modelo de prueba:

Este modelo describe principalmente cómo se prueban los componentes ejecutables (como las construcciones) en el modelo de implementación con pruebas de integración y de sistema. Describe además cómo deben probarse aspectos específicos del sistema (ejemplo interfaz de usuario, manual del usuario, etc. El modelo de pruebas es un conjunto de casos de prueba, procedimientos de prueba y componentes de prueba.

1.4.3.1.1 Casos de prueba:

Los casos de prueba especifican la forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse. Es un conjunto de entradas y resultados esperados que ejercitan a un componente con el propósito de causar fallas y detectar defectos. Entre los casos de prueba pueden existir todos los tipos de relaciones, pero las más importantes son las de precedencia.

Para obtener los casos de prueba hay que definir las condiciones de prueba (tabla 1.1). Las condiciones de prueba describen todas las situaciones que reflejen qué se quiere probar del sistema. Hay que priorizar las condiciones de prueba porque es posible que no se puedan probar todas o que algunas sean más críticas que otras.

¿De dónde se obtienen?	¿Cómo obtenerlas?
Casos de uso	Los diferentes cursos normales y alternativos.
Requerimientos funcionales	Debe hacer todo lo declarado.
Requerimientos no funcionales	Tiempos de respuesta esperados, volumen de información a manejar, configuración de hardware y software sobre los que debe ejecutarse, recuperación ante fallas, seguridad, ...
Especificaciones de diseño	Condiciones especiales para validar la integridad de módulos, implementaciones parciales si se va a hacer por etapas, ...

Tabla 1.1 Condiciones de prueba

Definición de los casos de prueba a partir de casos de uso

Esta propuesta desarrolla un método basado en tres pasos (tabla 1.2) para obtener un conjunto de casos de prueba del sistema a partir de casos de uso en UML.

Paso	Descripción	Resultado
1	Generar escenarios de uso	Todos los posibles caminos de ejecución de cada caso de uso. Cada camino es un escenario de uso.
2	Identificar casos de prueba	Conjunto de casos de prueba a partir de los escenarios anteriores.
3	Identificar los valores a probar	Valores de prueba asociados a cada caso de prueba anterior.

Tabla 1.2 Descripción de los pasos para la obtención de un conjunto de pruebas de sistema a partir de casos de uso

Para generar los casos de prueba primero se generan todos los posibles escenarios, o caminos de ejecución, de cada caso de uso. En el dibujo siguiente se muestra un ejemplo genérico donde se aprecia un camino de ejecución principal, flecha de color oscuro, y cuatro caminos de ejecución alternativos, uno que provoca la finalización de la ejecución y tres que hacen retroceder la ejecución a un paso anterior.

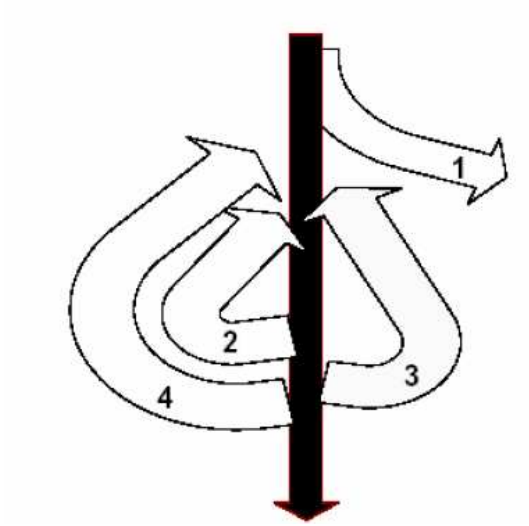


Figura 1.2 Ejemplo de un camino de ejecución principal y cuatro caminos alternativos

Después, se identifican los casos de prueba a partir de esos escenarios y, por último, se identifican los valores a probar de cada caso de prueba.

Tomando como punto de partida los casos de uso y su descripción no formal se obtiene al final una lista de casos de prueba, con los valores que deben probar y los resultados esperados para cada caso.

Los casos de pruebas deben verificar:

- Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificación de los requerimientos.

- Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

También, es necesario identificar casos de prueba que garanticen ejecutar pruebas de unidad, por lo general aplicando técnicas de prueba de caja blanca y pruebas de integración aplicando técnicas de prueba de caja negra si fuese necesario.

1.4.3.1.2 Procedimientos de prueba:

Los procedimientos de prueba especifican cómo realizar uno o varios casos de prueba o partes de estos, por lo general define los pasos a seguir por los probadores cuando tienen que verificar las condiciones de prueba incluidas en los casos de prueba.

Tienen que corresponderse con el flujo de eventos del caso de uso, pero dando valores.

1.4.3.1.3 Componentes de prueba:

Un componente de prueba automatiza uno o varios procedimientos de prueba o parte de ellos. Pueden ser programados o utilizar una herramienta de automatización de pruebas.

1.4.3.2 Plan de prueba:

Este describe la estrategia, recursos y cronograma de trabajo. La estrategia de prueba incluye la definición de tipos de prueba a realizar en la iteración, sus objetivos, el enfoque de las pruebas, entre otros.

1.4.3.3 Evaluación de prueba:

Es una evaluación de los resultados de los esfuerzos de prueba.

1.5 Buenas prácticas en el trabajo personal y en equipo

La calidad con la que se realiza un proceso de prueba no solamente está gobernada por la calidad de sus partes sino que cualquier error trivial en el trabajo realizado por parte del equipo de prueba o equipo de

desarrollo puede significar defectos en el producto. Por tanto es una necesidad que el trabajo de ambos equipos sea efectivo.

Para lograr que el trabajo sea efectivo se deben tener en cuenta tres aspectos que plantea el Proceso de Software Personal (PSP): [5]

1. Planificar su trabajo
2. Hacer su trabajo de acuerdo con el plan
3. Esforzarse en producir productos de alta calidad

En las actividades que propone el flujo de trabajo de prueba se debe incluir el PSP como una disciplina personal que desarrolla habilidades personales y muestra cómo aplicar métodos avanzados a cada actividad realizada para hacerlo de la mejor forma. Con el apoyo del PSP cada trabajador puede mejorar el rendimiento en su planificación y la calidad de cada actividad que realiza, en los primeros momentos de su utilización significará sólo un apoyo pero luego se comprueba cómo los procesos definidos guían el trabajo.

Una buena práctica en el flujo de trabajo de prueba es que los probadores utilicen y mantengan actualizado el Cuaderno del ingeniero, que contribuye entre otras cosas, a controlar el tiempo y lograr mejores planificaciones.

Establecer la calidad como máxima prioridad personal supone un considerable esfuerzo y perseverancia al aplicar el PSP y hacerlo de una forma consistente y consciente.

Es útil que en el proceso de prueba se proponga un esquema de trabajo donde cada trabajador tenga perfectamente definido sus roles, sus actividades, y sus responsabilidades y el equipo controle la ejecución de las tareas, tal y como lo plantea el Proceso de Desarrollo en Equipo (TSP), los roles distribuyen la gestión entre ingenieros, definen las responsabilidades para gestionar el entorno de trabajo. Además la existencia de roles y tareas bien definidas para cada uno de los integrantes contribuye

[5] Humphrey, W. S. Introducción al Proceso Software Personal. Madrid 2001

enormemente a que todos participen, y así se motiva y mantiene la disciplina de los miembros con el objetivo de mejorar la calidad del software producido y principalmente promover la integración del equipo de trabajo.

1.6 Tipos de pruebas

Cualquier producto de ingeniería puede ser probado de una de estas formas:

1. Conociendo la funcionalidad específica para la cual fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa.
2. Conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que "todas las piezas encajen", o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

El 1er enfoque se denomina Prueba de Caja Negra y el 2do Prueba de Caja Blanca.

1.6.1 Prueba de caja blanca:

Se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar.

1.6.1.1 Técnicas de pruebas de caja blanca

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que:[6]

[6] Igual a la Referencia 4

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada modulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

1.6.1.1.1 Prueba del camino básico

La prueba del camino básico es una técnica de prueba de Caja Blanca propuesta por Tom McCabe, y será utilizada en este Trabajo de Diploma para dar cumplimiento a uno de los objetivos que se proponen en el mismo.

Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico.

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

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

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.[7]

1. Notación de Grafo de Flujo

Para aplicar la técnica del camino básico se debe introducir una sencilla notación para la representación del flujo de control, el cual puede representarse por un Grafo de Flujo.

Cada nodo del grafo corresponde a una o más sentencias de código fuente. Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo.

Para construir el grafo se debe tener en cuenta la notación para las instrucciones. Figura 1.3 y Figura 1.4.

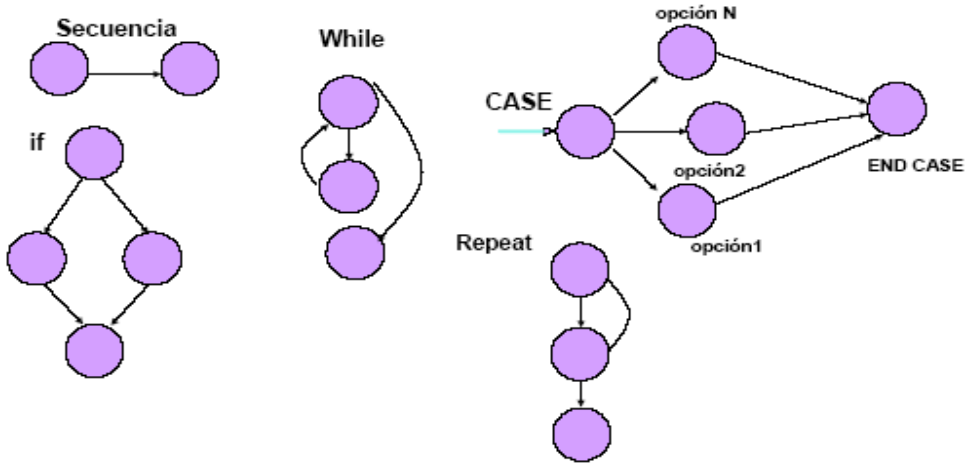


Figura 1.3 Notación de grafos de flujo para las instrucciones: Secuenciales, If, While

[7] Igual a Referencia 2

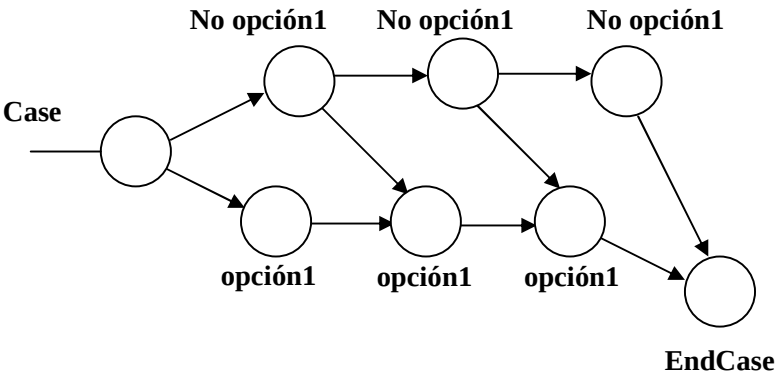


Figura 1.4 Notación de grafos de flujo para la instrucción *Case*

Un Grafo de Flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, comprensión y nos brinda información para confirmar que el trabajo se está haciendo adecuadamente.

Los componentes son:

1. Nodo.

Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que hallan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.

2. Aristas.

Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.

3. Regiones.

Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran y la cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.
Un ejemplo de representación de Grafo de Flujo a partir de un segmento de código del método BuscarHSF del módulo Registro de Población es el mostrado en la Figura 1.5 en el cual aparecen sus componentes:

```
1 require_once 'PLASER/Client.php';
1 require_once 'PLASER/dbz_class.php';
1 global $espacio;
2 if($plaser->get_nivel()!= 4)
3 $plaser->fault("No está autorizado a realizar búsquedas de HSF que no le pertenecen en su nivel",1);
4 $agente = new PLASER_Client();
4 $db = new DBz('m');
4 $num_hsf  = $args['num_hsf'];
4 $tipo_hsf = $args['tipo_hsf'];
4 $fecha_creacion = $args['fecha_creacion'];
4 $telefono = $args['telefono'];

5 if(is_array($args['id_casa']) and 6 count($args['id_casa']) > 0)
 {
       7 for ($i = 0; $i < count($args['id_casa']); $i++)
          8 $id_casa[$i] = $args['id_casa'][$i];
9 }
```

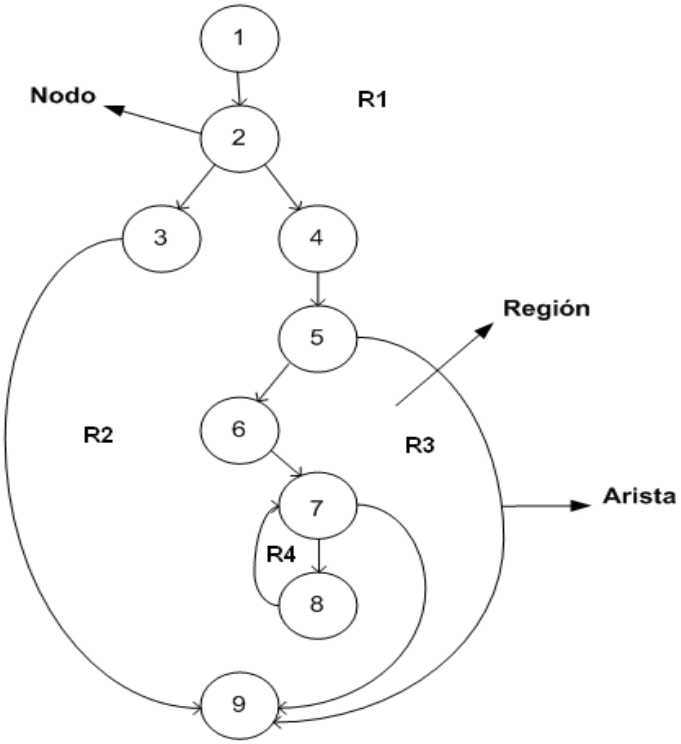


Figura 1.5 Características de los grafos.

Cualquier representación del diseño procedimental se puede traducir a un grafo de flujo. Cuando en un diseño se encuentran condiciones compuestas (uno o más operadores AND, NAND, NOR lógicos en una sentencia condicional), la generación del grafo de flujo se hace un poco más complicada.

2. Complejidad Ciclomática

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado, define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente.
La complejidad ciclomática de un grafo de flujo se puede calcular a través de 3 vías. Para el ejemplo mostrado en la figura 1.5 sería:

1ra vía V (G)= Número de Regiones = 4
2da vía V (G)= Número de Aristas – Número de Nodos + 2 = 11 – 9 + 2 = 4
3ra vía V (G)= Número de Nodos Predicados +1 = 3 + 1 = 4

3. Derivación de casos de prueba.

Luego de tener elaborados los Grafos de Flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino.

1.6.2 Prueba de caja negra:

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software.

1.6.2.1 Técnicas de pruebas de caja negra

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca.

Muchos autores, entre los cuales se puede mencionar a Pressman consideran que estas pruebas permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien.

Según Pressman para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

1. Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
2. Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
3. Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, 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 pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar.

1.6.2.1.1 Partición Equivalente

Una partición equivalente es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. [8]

Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

- Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
- Si una entrada es booleana, hay 2 clases: si o no.

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Aplicando estas directrices se ejecutan casos de pruebas para cada elemento de datos del campo de entrada a desarrollar. Los casos se seleccionan de forma que ejerciten el mayor número de atributos de cada clase de equivalencia a la vez.

[8] Igual a la Referencia 2

Para aplicar esta técnica de prueba se tienen en cuenta los siguientes pasos:

Primeramente se deben identificar las clases de equivalencia lo cual se hace tomando cada condición de entrada y aplicándole las directrices antes expuestas. (Ver Tabla 1.3)

Condición externa	Clases de equivalencia válidas	Clases de equivalencia Inválidas
Condición de entrada	Clases válidas para esa condición de entrada	Clases inválidas para esa condición de entrada

Tabla 1.3 Tipos de clases de equivalencia

Para definir las clases de equivalencia hace falta tener en cuenta un conjunto de reglas:

- Si una condición de entrada especifica un rango, entonces se confeccionan una clase de equivalencia válida y 2 inválidas.
- Si una condición de entrada especifica la cantidad de valores, identificar una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica un conjunto de valores de entrada y existen razones para creer que el programa trata en forma diferente a cada uno de ellos, identificar una clase válida para cada uno de ellos y una clase inválida.
- Si una condición de entrada especifica una situación de tipo “debe ser”, identificar una clase válida y una inválida.
- Si existe una razón para creer que el programa no trata de forma idéntica ciertos elementos pertenecientes a una clase, dividirla en clases de equivalencia menores.

Luego de tener las clases válidas e inválidas definidas, se procede a definir los casos de pruebas, pero para ello antes se debe haber asignado un identificador único a cada clase de equivalencia. Luego entonces se pueden definir los casos teniendo en cuenta lo siguiente:

a. Escribir un nuevo caso de cubra tantas clases de equivalencia válidas no cubiertas como sea posible hasta que todas las clases de equivalencia hayan sido cubiertas por casos de prueba.
b. Escribir un nuevo caso de prueba que cubra una y solo una clase de equivalencia inválida hasta que todas las clases de equivalencias inválidas hayan sido cubiertas por casos de pruebas.

Con la aplicación de esa técnica se obtiene un conjunto de pruebas que reduce el número de casos de pruebas y dicen algo sobre la presencia o ausencia de errores. A menudo se plantea que las pruebas a los software nunca terminan, simplemente se transfiere del desarrollador al cliente. Cada vez que el cliente usa el programa está llevando a cabo una prueba. Aplicando el diseño de casos de pruebas al software en cuestión se puede conseguir una prueba más completa y descubrir y corregir el mayor número de errores antes de que comiencen las “pruebas del cliente”.

1.7 Niveles o Estrategias de prueba

La Prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes niveles de pruebas:

- Prueba de unidad
- Prueba de integración
- Prueba de sistema
- Prueba de aceptación

1.7.1 Prueba de Unidad:

Es la prueba enfocada a los elementos testeables más pequeños del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca.

Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del componente. Si los datos no entran correctamente, todas las demás pruebas no tienen sentido.

El diseño de casos de prueba de una unidad comienza una vez que se ha desarrollado, revisado y verificado en su sintaxis el código a nivel fuente.

El tipo de prueba adecuada a este nivel es caja blanca.

1.7.2 Prueba de Integración:

Es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso. Se prueba un paquete o un conjunto de paquetes del modelo de implementación. Estas pruebas descubren errores o que las especificaciones de las interfaces de los paquetes están incompletas. Esta prueba debe ser responsabilidad de desarrolladores y de independientes, sin solaparse las pruebas.

Es el proceso de combinar y probar múltiples componentes juntos. El objetivo es tomar los componentes probados en unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.

Se llama integración incremental cuando el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir, es más probable que se pueda probar completamente las interfaces y se puede aplicar un enfoque de prueba sistemática.
Hay dos estrategias de integración incremental:

1. Integración Descendente (Top-Down):

Se integran los módulos moviéndose hacia abajo por la jerarquía de control. Comenzando por el módulo principal, los módulos subordinados se van incorporando a la estructura bien, en forma primero en profundidad, que integra todos los módulos de un camino de control principal de la estructura, o primero en anchura, que incorpora todos los módulos directamente subordinados a cada nivel, moviéndose por la estructura de forma horizontal.

Este proceso se realiza en una serie de cinco pasos:

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

El programa continúa desde el paso 2 hasta que se haya construido la estructura del programa entero.
Al aplicar esta estrategia pueden surgir algunos problemas, el más común se da cuando se requiere un proceso de los niveles más bajos de la jerarquía para poder probar adecuadamente los niveles superiores. Al principio de la prueba descendente, los módulos de bajo nivel se reemplazan por *resguardos*; por tanto, no pueden fluir datos significativos hacia arriba por la estructura del programa. Para solucionar esto se tienen tres opciones:

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

2. Integración Ascendente (Bottom-Up):

Empieza la construcción y la prueba con los módulos de los niveles más bajos de la estructura del programa. Dado que los módulos se integran de abajo hacia arriba, el proceso requerido de los módulos subordinados a un nivel dado siempre están disponibles y se elimina la necesidad de resguardos.

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

1. Se combinan los módulos de bajo nivel en grupos que realicen una subfunción específica del software.
2. Se escribe un controlador para coordinar la entrada y la salida de los casos de prueba.

3. Se prueba el grupo.
4. Se eliminan los controladores y se combinan los grupos moviéndose hacia arriba por la estructura del programa.

A medida que la integración progresa disminuye la necesidad de controladores de prueba diferentes.

La selección de una estrategia de integración depende de las características del software y de la planificación del proyecto.

Una buena alternativa es usar una mezcla de las dos estrategias (Ascendente y Descendente) que use la descendente para los niveles superiores de la estructura, junto con la ascendente para los niveles subordinados.

A medida que progresa la prueba de integración, se deben identificar los módulos críticos. Un módulo crítico es aquel que tiene las una o más de las siguientes características:

- Está dirigido a varios requisitos del software
- Tiene un mayor nivel de control
- Es complejo o propenso a errores
- Tiene unos requisitos de rendimiento muy definidos
- Los módulos críticos deben probarse lo antes posible.

Diseño de casos de pruebas de integración.

Deben ser diseñados los casos de prueba de integración para verificar que los componentes interaccionan entre sí de la forma apropiada después de ser integrados en una construcción. Estos casos de prueba pueden ser derivados de las realizaciones de los casos de uso de diseño (un diagrama de secuencia es parte de la realización de caso de uso diseño) o los diagramas de interacción de las realizaciones de casos de uso ya que estas realizaciones describen cómo interaccionan las clases y los objetos, y por tanto cómo interaccionan los componentes.

Los diseñadores de pruebas buscan combinaciones de entradas, salida y estado inicial de sistema que den lugar a escenarios interesantes que empleen las clases (y por tanto los componentes) que participan en los diagramas.

En el caso de integrar varios módulos y se encuentra un error en el momento de integrarlos, se tiene que hacer una Prueba de Regresión**.**

Prueba de Regresión

Cada vez que se añade un nuevo módulo como parte de una prueba de integración, el software cambia. Estos cambios pueden causar problemas con funciones que antes trabajaban perfectamente. La prueba de regresión es la actividad que ayuda a asegurar que los cambios no introducen un comportamiento no deseado o errores adicionales. El conjunto de pruebas de regresión contiene tres clases diferentes de casos de prueba:

- Una muestra representativa de pruebas que ejercite todas las funciones del software.
- Pruebas adicionales que se centran en las funciones del software que se van a ver probablemente afectadas por el cambio.
- Pruebas que se centran en los componentes del software que ha cambiado.

Diseño de casos de prueba de regresión.

Algunos de los casos de pruebas pueden ser usados para pruebas de regresión en construcciones subsiguientes, aunque no todos pueden ser utilizados para pruebas de regresión. Los casos de pruebas, para ser usados en pruebas de regresión, deben ser suficientemente flexibles para ser resistentes a cambios, esta flexibilidad debe ser cuidadosa ya que la conversión de un caso de prueba a caso de prueba de regresión supone un esfuerzo de desarrollo extra, luego se debe convertir casos de prueba a casos de prueba de regresión sólo cuando el esfuerzo merezca la pena.

En la fase de construcción cuando el grueso del sistema está implementado se llevan a cabo los niveles de prueba: Sistema y Aceptación.

1.7.3 Prueba de Sistema:

Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados.

En un ciclo iterativo estas pruebas ocurren más temprano, tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados.

El tipo de prueba adecuada a este nivel es caja negra.

1.7.3.1 Tipos de Pruebas del Sistema

Prueba de Recuperación: Es una prueba del sistema que fuerza el fallo del software de muchas formas y verifica que la recuperación se lleva a cabo apropiadamente.

Prueba de Seguridad: Intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán, de hecho, de acceso impropios.

Prueba de Resistencia: Están diseñadas para enfrentar a los programas con situaciones anormales.

Prueba de Rendimiento: Está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado.

Prueba de funcionalidad: Puede estar orientada a probar específicamente la función, que es cuando se fija la atención en la validación de las funciones, métodos, servicios casos de uso etc. También puede tratar de buscar la seguridad asegurando que los datos o el sistema solamente son accedidos por actores deseados. O para probar el volumen, verificando las habilidades de los programas para manejar grandes cantidades de datos.

Prueba de Stress: Enfocadas a evaluar cómo el sistema responde bajo las condiciones anormales. Ejemplo: extrema sobrecarga, insuficiente memoria, servicios y hardware, no disponibles o recursos compartidos no disponibles etc.

Prueba de usabilidad: Puede medir la interfaz de usuario, el uso de la misma, la salida de los programas, la comprensión de los mensajes de la aplicación, la integridad conceptual de las interfaces, el número excesivo de opciones etc.

Prueba de configuración: Enfocadas a asegurar que funcione en diferentes configuraciones de software y hardware, es implementada también como prueba de rendimiento del sistema.

Prueba de instalación: Enfocadas a garantizar la instalación en diferentes configuraciones de hardware y software, bajo diferentes condiciones, puede ser bajo insuficiente espacio en disco u otras.

Diseño de los casos de prueba de sistema.

Las pruebas de sistema se usan para probar que el sistema funciona correctamente como un todo. Cada prueba de sistema prueba principalmente combinaciones de casos de uso instanciados bajo condiciones diferentes. Estas condiciones incluyen diferentes configuraciones hardware, diferentes niveles de carga del sistema, diferentes números de actores y diferentes tamaños de las bases de datos. Cuando se desarrollan los casos de prueba de sistema los diseñadores de prueba deberían dar prioridad a las combinaciones de los casos de uso que:

- Se necesita que funcionen en paralelo
- Posiblemente funcionan en paralelo
- Posiblemente se influencian mutuamente si se ejecutan en paralelo
- Involucran varios procesadores
- Usa frecuentemente recursos del sistema, como procesos, procesadores, bases de datos y software de comunicaciones, quizás en formas complejas e impredecibles.

1.7.4 Prueba de Aceptación:

Prueba de aceptación del usuario es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Con la realización de este capítulo se puede arribar a conclusiones relacionadas con las estrategias a seguir para llevar a cabo el proceso de prueba a un software, tales como: aunque se debe aplicar durante todo el proceso de desarrollo del software tiene mayor importancia en las fases de elaboración y construcción. En la fase de elaboración se implementan los niveles de unidad e integración, y necesitan tipos de prueba específicos para cada uno , existiendo además técnicas a emplear para cada tipo de prueba. Y en la fase de construcción los niveles son sistema y aceptación, con un tipo de prueba para este nivel y técnicas propias de este tipo de prueba. Y que como premisa fundamental se debe lograr un producto con calidad para lo cual es imprescindible aplicar las buenas prácticas de Ingeniería de Software en cada actividad de este flujo de trabajo así como una disciplina personal y en equipo que haga del trabajo el más óptimo.

CAPÍTULO
PLAN DE PRUEBA

En este capítulo se definirá el plan de pruebas diseñado y utilizado para guiar el proceso de prueba de la primera iteración de la fase de construcción de la opción HSF del módulo RPOB detallando:

- Propósito
- Alcance
- Estrategia de evolución del Plan
- Requerimientos a probar
- Estrategia de prueba
- Tipos de prueba
- Infraestructura de pruebas
- Recursos

Plan de Prueba del módulo RPOB

2.1 Introducción

Este documento se confecciona con el objetivo de definir el Plan de pruebas para la aplicación RPOB del Proyecto SISalud en la primera iteración de la fase de construcción del primer ciclo de desarrollo del módulo. Se especificará la estrategia de prueba para dicha iteración así como la estrategia de evolución del plan, infraestructura de prueba, recursos entre otros.

2.2 Propósito

Este Plan de Prueba para el módulo RPOB soporta los siguientes objetivos:
Definir el alcance por etapas de las pruebas de la iteración.

Definir la estrategia de evolución de chequeo de este plan.
Definir la estrategia de pruebas en la iteración y como parte de esta el script de datos de prueba y el componente de prueba a diseñar.
Definir la infraestructura de prueba.
Definir los recursos necesarios.
Definir los roles y las responsabilidades.
Definir el cronograma de trabajo.

2.3 Alcance

Se propone las siguientes etapas para la realización de las pruebas

1. **Etapa I**. Pruebas de unidad a la funcionalidad Insertar Problemas de Salud, incluida dentro de la opción HSF.
2. **Etapa II**. Pruebas de integración a la opción HSF de la aplicación con las funcionalidades que engloba esta opción pues en estos momentos es la de mayor prioridad y por tanto está en un nivel más avanzado en la fase construcción.

2.4 Estrategia de evolución del Plan

El plan será chequeado diariamente verificando si es necesario algún cambio. Al finalizar el día se reúne el equipo de prueba para verificar el cumplimiento del plan en cuanto al cronograma, analizar los cambios e informar al equipo de desarrollo de los resultados obtenidos para facilitar el proceso de cambio en paralelo al proceso de prueba.

2.5 Requerimientos para probar

Requerimientos funcionales y no funcionales que forman parte de la documentación del proyecto.

2.6 Estrategia de prueba

A continuación se detalla el flujo de trabajo prueba que se llevará a cabo y las diferentes estrategias que en cada una de las etapas.

De manera general el flujo de prueba inicia cuando el grupo de prueba define el plan de prueba y comienza las actividades propias del proceso de prueba.

Para la etapa I se definirán y especificarán casos de prueba de unidad para la funcionalidad Insertar problemas de Salud, mediante el tipo de prueba de caja blanca y utilizando la técnica del camino básico que comprueba la lógica interna de cada instancia. Se ejecutarán los casos de prueba definidos mediante un componente de prueba que es necesario desarrollar, del cual se describe la estrategia de cómo diseñarlo e implementarlo en el epígrafe: Componente de prueba y que debe automatizar este proceso permitiendo según determinados juegos de datos la obtención de diferentes resultados de forma visual. Estos resultados serán evaluados en dependencia a los resultados esperados, en la medida que empiecen a aparecer errores estos serán anotados para al terminar el día elaborar una lista de defectos, que se le entregará al líder de proyecto por parte de la UCI, quien informará de estos errores al equipo de desarrollo motivando un proceso de gestión de cambios paralelo con el proceso de prueba

Para la etapa II se definirán casos de prueba de caja negra utilizando la técnica de partición equivalente para probar según los datos introducidos en los campos de entrada, ya sean válidos o inválidos, previamente especificados en los casos de prueba, los resultados. Estos casos de prueba se ejecutarán a través de la interfaz implementada para la aplicación de RPOB, comprobando la correcta integración de la funcionalidad con los correspondientes componentes de la capa de presentación y verificando la correspondencia con los resultados esperados.

A partir de aquí se llevará cabo un proceso similar a la etapa anterior, en la medida que empiecen a aparecer errores estos serán anotados para al terminar el día elaborar una lista de defectos, para lo cual se recomienda utilizar el cuaderno del ingeniero para ir archivando los defectos resultantes de la ejecución de la pruebas de forma organizada, esta lista de defectos se le entregará al líder de proyecto por parte de la UCI, quien informará de estos errores al equipo de desarrollo motivando un proceso de gestión de cambios paralelo con el proceso de prueba.

Para el cumplimiento de ambas etapas es de vital importancia definir los scripts de datos de prueba para cada uno de los componentes del proyecto SiSalud que se relacionan con el módulo RPOB teniendo en cuenta la relación de dependencia que existe entre los mismos, manifestándose un flujo de información que en este caso sólo se considera de otros módulos hacia el módulo RPOB. (Ver Figura 2.1)

Estos módulos son:

RAS (Registro de Áreas de Salud), RC (Registro de Ciudadano), RPSAP (Registro de Problemas de Salud para la Atención Primaria) y SAAA(Seguridad basada en el modelo de Autenticación, Autorización y Auditoría) partiendo de que ya existen los scripts de datos de prueba para los componentes RUS (Registro de Unidades de Salud), RU(Registro de Ubicación), RL(Registro de Localidades), RPS(Registro de Personal de la Salud), RE(Registro de Estudiante), RSM(Registro de Servicios Médicos), REDO(Registro de Enfermedades de Declaración Obligatoria) y RCIE(Registro de Clasificación Internacional de Enfermedades)

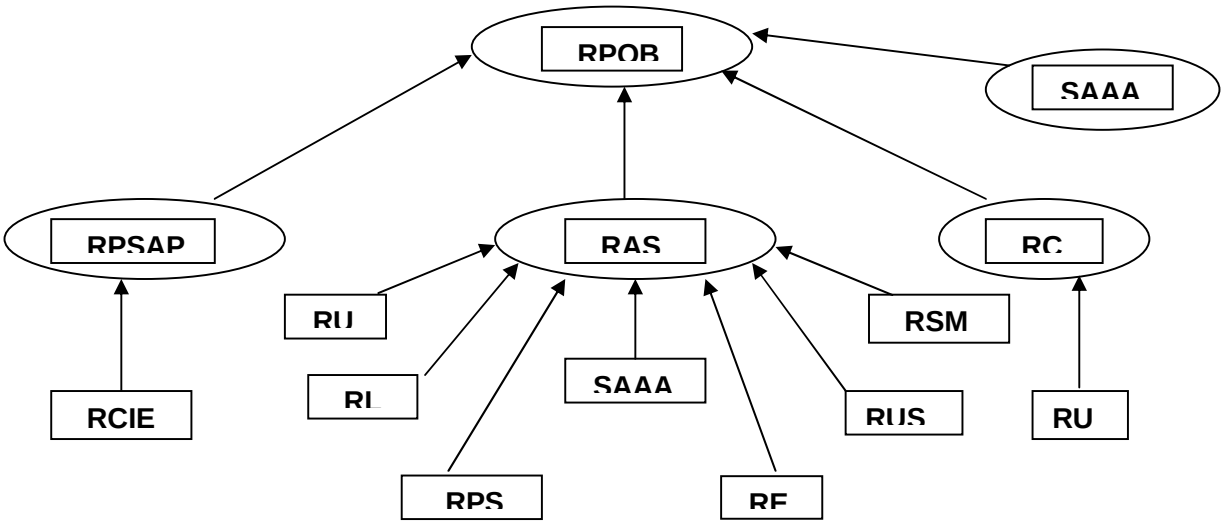


Figura 2.1 Flujo de información entre módulos de SISalud

En la siguiente figura se muestra el proceso de prueba que se propone:

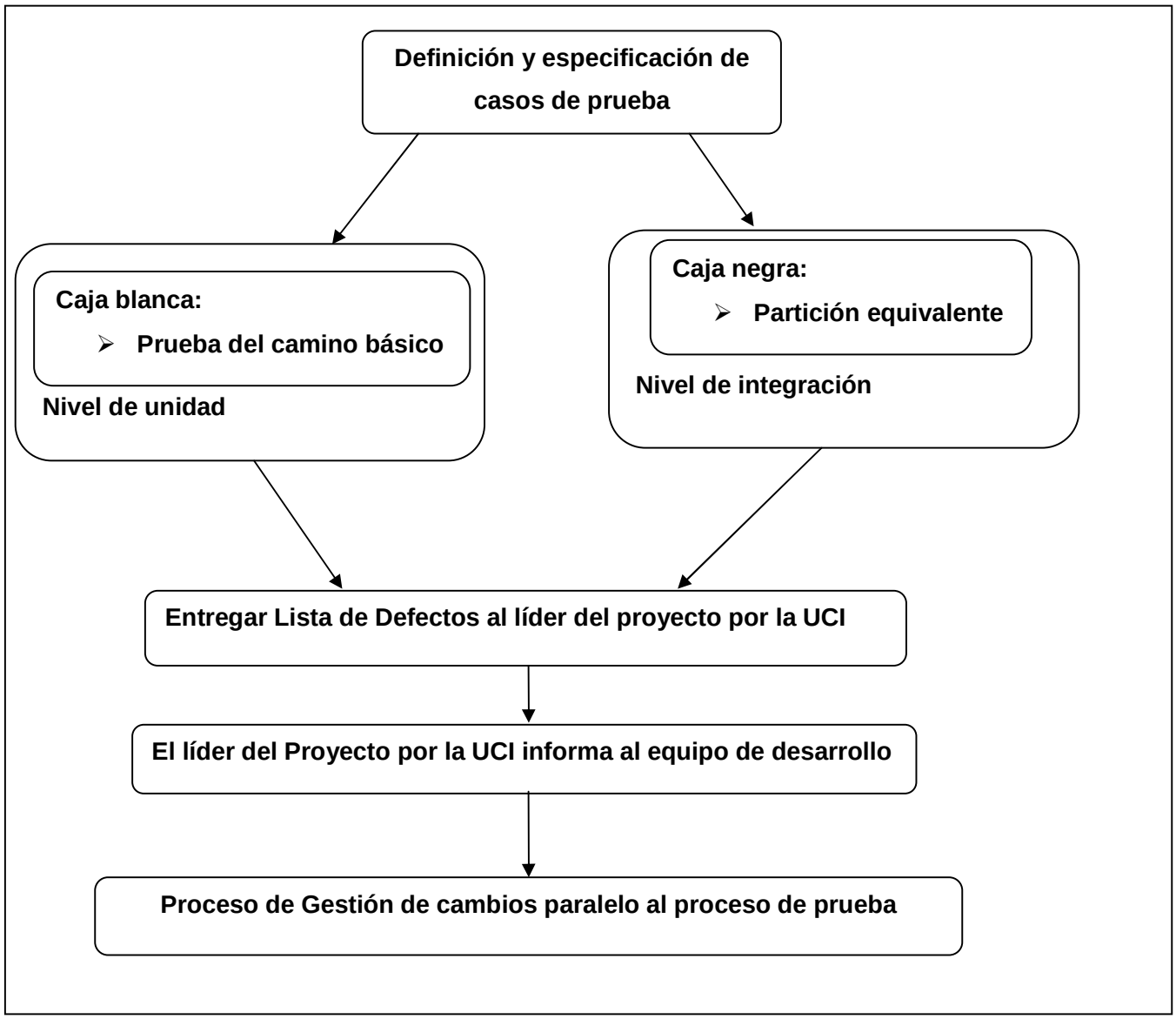


Figura 2.2 Proceso de prueba

2.7 Componente de prueba

Para el cumplimiento de la etapa I, dentro de la ejecución de los casos de prueba de caja blanca se debe generar un componente de prueba que automatice el proceso de ejecución de los mismos. Para ello se debe añadir un nuevo componente en el RIS (Registro Informatizado de Salud), con el nombre de Componente de prueba, se debe realizar el XSL y un fichero php de capa de presentación que es el cliente para implementar el funcionamiento del componente en la interfaz desde donde se van a ejecutar los casos de prueba.

Se debe agregar en la tabla módulos de la Base de datos SAAA el nombre del componente de prueba y en la tabla usuario, un usuario para acceder al RIS, con derecho de editor perteneciente al nivel nacional. Se deben crear los host virtuales para configurar la dirección que va a acceder al componente y se debe construir el fichero XML para crear la interfaz donde se va a mostrar el componente.

2.8 Tipos de prueba:

2.8.1 Pruebas de Caja Blanca

Al método Insertar Problemas de Salud de la capa de negocio.

2.8.2 Pruebas de Caja Negra

Deben ser probadas las funcionalidades:

- ✓ Nueva HSF
- ✓ Buscar HSF
- ✓ Buscar Integrante

Además al probar cada funcionalidad anteriormente descrita deben realizarse:

- ➢ Pruebas de seguridad y control de acceso

- Pruebas de integridad de los datos a la Base de datos
- Pruebas de Interfaz de Usuario.

2.8.2.1 Pruebas de Seguridad y Control de Acceso

La Prueba de Seguridad y Control de Acceso se enfoca en:
Seguridad en el ámbito de aplicación, incluyendo el acceso a los datos y a las funciones de negocios.
La seguridad en el ámbito de aplicación asegura que, basado en la seguridad deseada los usuarios están restringidos a funciones o casos de uso específicos o limitados en los datos que están disponibles para ellos.
Se deben crear usuarios de tipo visualizador y editor, ambos pertenecientes al nivel de Unidad de Salud, el cual tiene acceso a la opción HSF del módulo RPOB.

Objetivo de la prueba

Seguridad en el ámbito de aplicación: Verificar que un usuario pueda acceder solo a las funciones o datos para los cuales su tipo de usuario tiene permiso.

Los usuarios de tipo visualizador del nivel Unidad de Salud tendrán acceso sólo a las búsquedas de HSF e Integrante.

Los usuarios de tipo editor del nivel Unidad de Salud tendrán acceso a todas las funcionalidades incluidas en la opción HSF.

Técnica

Seguridad en el ámbito de aplicación: Identificar y hacer una lista de cada tipo de usuario y las funciones y datos sobre las que cada tipo tiene permiso. (Ver Tabla 2.1)

Crear pruebas para cada tipo de usuario y verificar cada permiso creando operaciones específicas para cada tipo de usuario.

Modificar el tipo de usuario y volver a ejecutar las pruebas para los mismos usuarios. En cada caso, verificar que las funciones o datos adicionales están correctamente disponibles o son denegados.

Usuario	Contraseña	Tipo	Nivel
Darlen07	darlen2007	Visualizador	Unidad de Salud
Yisel07	yisel2007	Editor	Unidad de Salud

Tabla 2.1 Lista de tipos de usuarios

Criterio de aceptación

Para cada tipo de usuario conocido las funciones y datos apropiados están disponibles, y todas las operaciones funcionan como se espera y ejecutan las pruebas de Funcionalidad de la aplicación.

2.8.2.2 Prueba de integridad de los datos y la base de datos

Objetivo de la prueba

El objetivo de esta prueba es verificar que las inserciones, actualizaciones, y eliminaciones de la BD se ejecutan debidamente, es decir si los métodos de estas funcionalidades funcionan bien.

Técnica

Se incorporarán datos de pruebas en la BD utilizando el script de datos de prueba, para cuando se ejecute un método listar estos sean los datos que se deban mostrar. Se pasarán por parámetros datos de pruebas que una vez ejecutados los métodos listar puedan verificar estos. Además se inspeccionará la BD para asegurarse que los datos se han guardado o actualizado correctamente.

Criterio de aceptación

Todos los métodos y procesos de acceso a la base de datos funcionan como fueron diseñados y sin datos corruptos.

Consideraciones especiales

Se ejecutarán los métodos a través de la capa de presentación y con usuarios (Ver Tabla 2.1) con el debido acceso para facilitar la prueba, las mismas deberán devolver datos de pruebas insertados en la BD con anterioridad.

2.8.2.3 Prueba de Interfaz de Usuario

Esta prueba verifica que la interfaz de la aplicación proporcione al usuario el acceso y navegación a través de las funciones apropiadas. Además asegura que los objetos presentes en la interfaz de usuario se muestren como se espera y conforme a los estándares establecidos por el proyecto.

Se verificará lo siguiente:

Navegación a través de todas las funcionalidades, verificar que cada interfaz es amigable al usuario.
Verificar las funciones de Ayuda Online.

Objetivo de la prueba

Verificar que la navegación a través de los elementos que se están probando reflejen las funciones del negocio y los requerimientos solicitados, de acuerdo a los estándares establecidos, incluyendo las validaciones de los campos, métodos de acceso como los menús y las opciones de ayuda.
Técnica

Crear o modificar pruebas para cada tipo de usuario verificando la navegación y los estados de los objetos para cada interfaz y cada objeto contemplado dentro de la misma.
Criterio de aceptación
Cada ventana ha sido verificada exitosamente siendo consistente con una versión de referencia o estándar establecido.

Consideraciones especiales

Se ejecutaran los métodos a través de la capa de presentación y con usuarios con el debido acceso (Ver Tabla 2.1) para facilitar la prueba.

2.9 Infraestructura de Pruebas

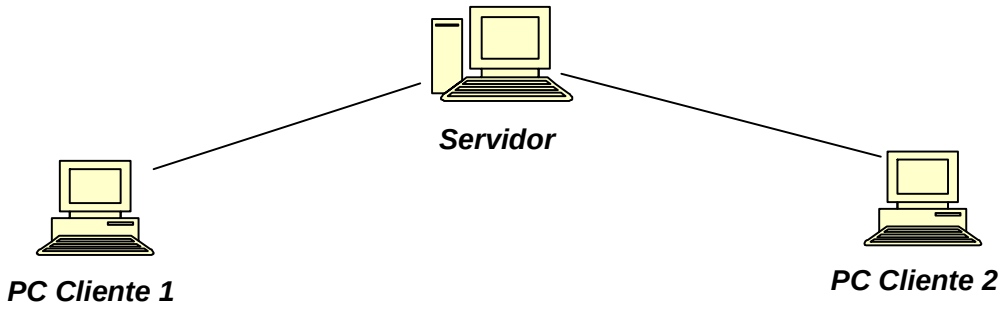


Figura 2.3 Infraestructura de Pruebas

Escenario de Pruebas	
Servidor	Pentium IV 3,00 GHz, 2 Gb RAM y 250 Gb de Disco Duro.
Cliente 1 (Administrador de Prueba, Analista de Prueba y Probador)	Pentium IV 2.40 GHz, 248 MB RAM y 80 Gb de Disco Duro.
Cliente 2 (Diseñador de Prueba y Probador)	Pentium IV 2.40 GHz, 248 MB RAM y 80 Gb de Disco Duro.

Tabla 2. 2 Requerimientos de hardware

Escenario de Pruebas	
Servidor	MySQL-Front 3.2
Cliente 1 (Administrador de Prueba, Analista de Prueba y Probador)	MySQL-Front 3.2 Servidor HTTP (preferiblemente Apache) que soporte VirtualHosts Navegadores: Mozila o Internet Explorer
Cliente 2 (Diseñador de Prueba y Probador)	PHP 4.3.4 MySQL-Front 3.2 Servidor HTTP (preferiblemente Apache) que soporte VirtualHosts Navegadores: Mozila o Internet Explorer Dreamweaver 8 Stylus Studio 5 Nusphere 4.0

Tabla 2.3 Requerimientos de software

2.9.1 Recursos

Los recursos sobre los cuales se ejecutarán las pruebas se propone que sean 3 PC, una que funcionará como servidor y 2 como clientes, con un equipo de pruebas integrado por 2 estudiantes que desempeñarán todos los roles partiendo de que tienen que ejecutar todas las actividades del flujo de trabajo de prueba y guiados por los lideres del proyecto de la UCI y SOFTEL respectivamente.

2.10 Roles y responsabilidades

Rol	Cantidad	Responsabilidades
Administrador de Prueba	1	➢ Genera el Plan de prueba para la iteración. ➢ Negociar o acordar las pruebas. ➢ Verifica el progreso y efectividad de las pruebas ➢ Genera la Evaluación de los resultados
Analista de Prueba	1	➢ Participa en la Evaluación de los resultados ➢ Participa en la elaboración del Plan de Prueba ➢ Identifica los objetivos de la pruebas ➢ Identifica las pruebas requeridas para cada iteración
Diseñador	1	➢ Identifica los casos prueba. ➢ Genera la descripción de los casos de prueba. ➢ Es responsable de la integridad del Modelo de prueba ➢ Implementa los Componentes de Prueba ➢ Genera el Script de datos de prueba ➢ Participa en la Evaluación de los resultados ➢ Participa en la elaboración del Plan de prueba
Probador	2	➢ Ejecuta las pruebas ➢ Elabora la Lista de Defectos ➢ Participa en la Evaluación de los resultados

		➢ Participa en la elaboración del Plan de Prueba
Líder de proyecto por la UCI	1	➢ Responsable del desarrollo de las pruebas. ➢ Responsable de revisiones técnicas formales al desarrollo de las pruebas. ➢ Responsable de aceptar o no el producto después de la fase de prueba.
Líder de Proyecto por SOFTEL	1	➢ Responsable del desarrollo de las pruebas. ➢ Responsable de revisiones técnicas formales al desarrollo de las pruebas. ➢ Responsable de aceptar o no el producto después de la fase de prueba.

Tabla 2.4 Roles y responsabilidades del Flujo de Pruebas

La planificación de los procesos de prueba es una etapa de vital importancia, ya que además de definir la estrategia y establecer los recursos necesarios, permite realizar estimaciones que establecen una base para el control y seguimiento durante la ejecución del proceso. Establece un referente para futuros proyectos y además ofrece información para realizar el análisis de los procesos para identificar aspectos de mejoramiento. La construcción de un buen plan de prueba es la piedra angular y en consecuencia uno de los principales factores críticos de un proceso de prueba que permite entregar un software de mejor nivel. Es importante no hacer los planes de prueba suponiendo que prácticamente no hay defectos en los programas, y por lo tanto dedicando pocos recursos a las pruebas, pues siempre hay defectos. [9]

[9] Gonzalez,C. Un Plan de Pruebas Exitoso.2002 [Disponible en http://www.americaxxi.cl/modules.php?name=News&file=article&sid=20].

CAPÍTULO

DISEÑO, EJECUCIÓN Y EVALUACIÓN DE LAS PRUEBAS

En este capitulo se definirán y especificarán los casos de prueba de caja blanca para la opción Insertar Problemas de Salud y los casos de prueba de caja negra para las funcionalidades que se incluyen en la opción HSF. Se evaluarán los resultados que se obtengan de la ejecución de los casos de prueba y se hará un resumen de su evaluación.

Luego de un estudio profundo, al hacer el análisis de las estrategias de prueba, incluida en el Plan de prueba, para las funcionalidades en la opción HSF, se ha decidido demostrar el proceso de ejecución de las pruebas de caja blanca a la funcionalidad Insertar Problemas de Salud por englobar la mayor complejidad. Se empleará la plantilla de especificación de casos de prueba que se utiliza en el proyecto SISalud y la técnica de prueba del Camino Básico. Para ello se genera el grafo de flujo que se muestra en la figura 3.1, se calcula la complejidad ciclomática que determina la cantidad de caminos básicos, que son la entrada para la definición de los casos de prueba.

3.1 Definición de los casos de prueba de caja blanca

Complejidad ciclomática de Insertar Problemas de Salud

V(G)=Número de regiones

= 22

V(G)=Número de aristas -número de nodos +2

= 77 - 57 + 2

=22

V(G)=Número de Nodos Predicados +1

=21 + 1

=22

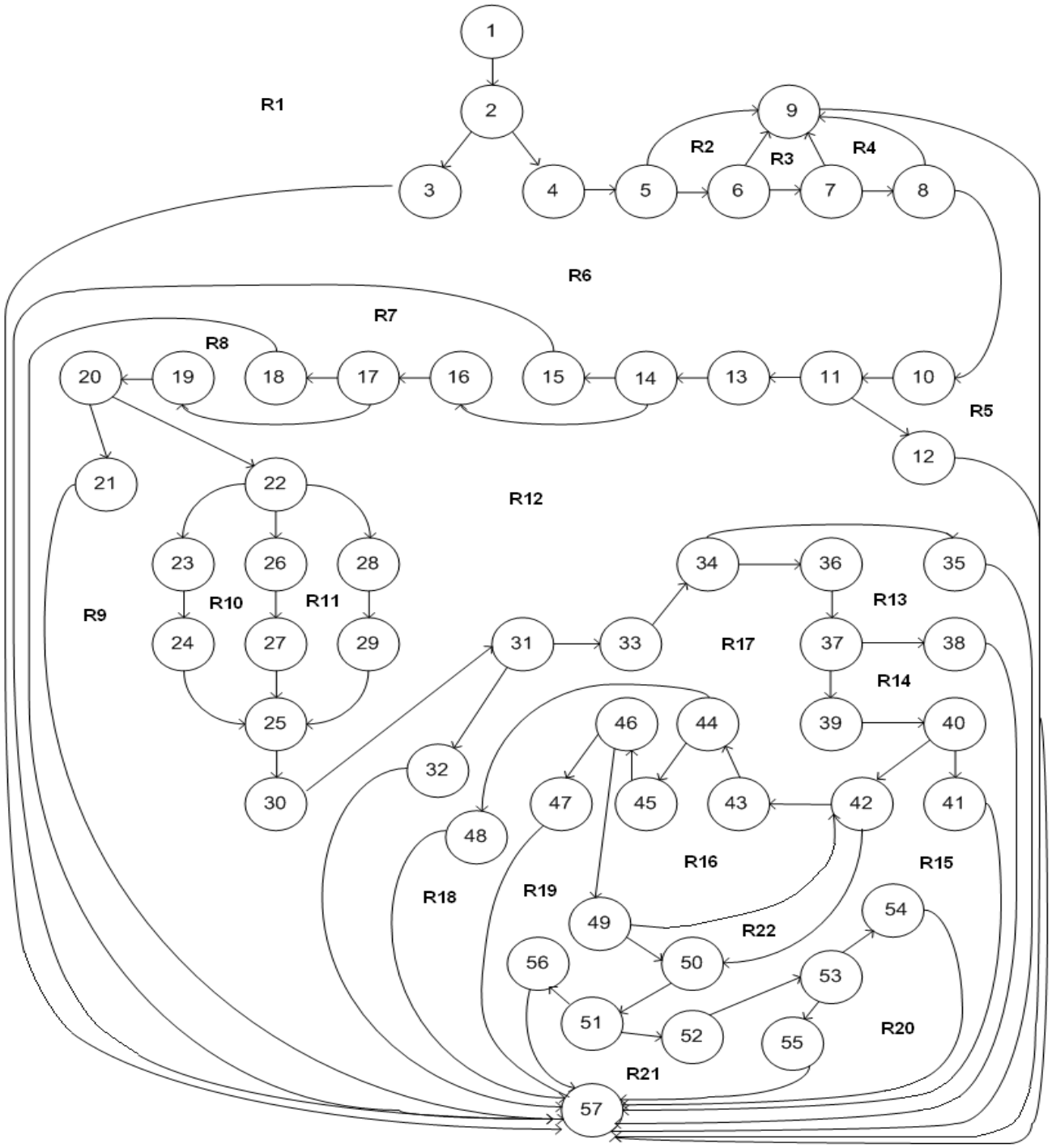


Figura 3.1.Grafo de Flujo para el método InsertarProblemaSalud

Caminos Básicos

CB 1

1-2-3-57

CB 2

1-2-4-5-6-7-8-9-57

CB 3

1-2-4-5-9-57

CB 4

1-2-4-5-6-9-57

CB 5

1-2-4-5-6-7-9-57

CB 6

1-2-4-5-6-7-8-10-11-12-57

CB 7

1-2-4-5-6-7-8-10-11-13-14-15-57

CB 8

1-2-4-5-6-7-8-10-11-13-14-16-17-18-57

CB 9

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-21-57

CB 10

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-24-25-30-31-32-57

CB 11

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-26-27-25-30-31-32-57

CB 12

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-28-29-25-30-31-32-57

CB 13

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-24-25-30-31-33-34-35-57

CB 14

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-24-25-30-31-33-34-36-37-38-57

CB 15

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-24-25-30-31-33-34-36-37-39-40-41-57

CB 16

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-24-25-30-31-33-34-36-37-39-40-42-43-44-45-46-47-57

CB 17

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-24-25-30-31-33-34-36-37-39-40-42-50-51-52-53-54-57

CB 18

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-24-25-30-31-33-34-36-37-39-40-42-43-44-48-57

CB 19

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-24-25-30-31-33-34-36-37-39-40-42-43-44-45-46-49-42-43-44-45-46-47-57

CB 20

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-24-25-30-31-33-34-36-37-39-40-42-50-51-56-57

CB 21

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-24-25-30-31-33-34-36-37-39-40-42-50-51-53-55-57

CB 22

1-2-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-24-25-30-31-33-34-36-37-39-40-42-43-44-45-46-49-50-51-52-53-54-57

3.2 Especificación de los casos de prueba de caja blanca

3.2.1 Caso de prueba Insertar Problemas de Salud

Descripción

Este caso de prueba cubre todo el proceso de inserción de uno o varios problemas de salud y este a su vez actualiza la dispensarización del paciente asignándole el grupo dispensarial al que pertenece en caso de cambiar, luego de haber adicionado algún problema de salud.

Las instancias en este caso de prueba son:

Caso de prueba correspondiente al CB 6
Caso de prueba correspondiente al CB 7
Caso de prueba correspondiente al CB 9
Caso de prueba correspondiente al CB 18
Caso de prueba correspondiente al CB 20
Caso de prueba correspondiente al CB 21

3.2.1.1 Caso de prueba correspondiente al CB 6

Descripción

En esta instancia se valida que los datos de entrada no contengan parámetros vacíos.

Condiciones de ejecución

Debe haber al menos un parámetro de entrada vacío.

Entrada

codigo_problema_salud = 32Q
tipo_problema = ‘Riesgo‘
fecha_deteccion = Vacío
id_ciudadano = ‘5‘

Resultados esperados

Muestra un mensaje: “FALTA UN PARÁMETRO POR PASAR O HA PASADO CERO EN ALGÚN PARÁMETRO”.

3.2.1.3 Caso de prueba correspondiente al CB 7

Descripción

En esta instancia se valida que la cadena de los datos de entrada no contenga caracteres extraños.

Condiciones de ejecución

Debe haber al menos un carácter extraño en alguno de los parámetros.

Entrada

codigo_problema_salud = ‘20Q ‘
tipo_problema = ‘Riesgo‘
fecha_deteccion = ‘2007-10-10‘
id_ciudadano = ‘(5) ‘

Resultados esperados

Muestra un mensaje: “ALGÚN PARÁMETRO INTRODUCIDO CONTIENE CARACTERES EXTRAÑOS”.

3.2.1.4. Caso de prueba correspondiente al CB 9

Descripción

En esta instancia se validan los datos que deben ser enteros positivos

Condiciones de ejecución

Debe haber un parámetro de entrada cuyo valor debe ser entero positivo y no lo sea.

Entrada

codigo_problema_salud = ‘5 ‘
tipo_problema = ‘Riesgo‘
fecha_deteccion = ‘2007-10-10‘
id_ciudadano = ‘4.5‘

Resultado esperado

Muestra un mensaje: “ESTE PÁRAMETRO DEBE SER ENTERO POSITIVO”.

3.2.1.5 Caso de prueba correspondiente al CB 18

Descripción

En esta instancia se inserta un problema de salud cuyo grupo dispensarial es mayor que el grupo dispensarial que tenía el paciente antes de insertarle el nuevo problema de salud. Además se actualiza el grupo dispensarial.

Condiciones de ejecución

La Base de datos debe tener información, el problema de salud que se desea insertar no debe existir y el grupo dispensarial asociado al problema de salud que se inserta debe ser mayor que el que tenía antes.

Entrada

codigo_problema_salud = ‘Q40 ‘
tipo_problema = ‘Riesgo‘
fecha_deteccion = ‘2007-10-10‘
id_ciudadano = ‘2‘

Resultado esperado

Se inserta el problema de salud en la tabla tb_problemas_salud_paciente con los valores de los parámetros que se especifican en la entrada.

Se actualiza en la tabla tb_paciente el id_grupo_dispensarial con el valor del nuevo problema de salud.

3.2.1.6 Caso de prueba correspondiente al CB 20

Descripción

En esta instancia se inserta un problema de salud cuyo grupo dispensarial es menor que el grupo dispensarial que tenía antes de la inserción.

Condiciones de ejecución

La Base de datos debe tener información, el problema de salud que se desea insertar no debe existir y el grupo dispensarial asociado al problema de salud que se inserta debe ser menor que el que tenía antes.

Entrada

codigo_problema_salud = ‘47‘
tipo_problema = ‘Riesgo‘
fecha_deteccion = ‘2007-10-10‘
id_ciudadano = ‘3‘

Resultado esperado

Se inserta el problema de salud en la tabla tb_problemas_salud_paciente con los valores de los parámetros que se especifican en la entrada.

No se modifica en la tabla tb_paciente el id_grupo_dispensarial.

3.2.1.7 Caso de prueba correspondiente al CB 21

Descripción

En esta instancia se intenta insertar un problema de salud que ya existe en un paciente.

Condiciones de ejecución

La Base de datos debe tener información y el problema de salud que se desea insertar debe existir asociado al paciente.

Entrada

codigo_problema_salud = ‘47‘
tipo_problema = ‘Riesgo‘
fecha_deteccion = ‘2007-10-10‘
id_ciudadano = ‘3‘

Resultado esperado

Muestra un mensaje: “YA EXISTE ESE PROBLEMA DE SALUD”

3.3 Definición y especificación de los casos de prueba de caja negra

Para todos los casos de prueba que a continuación se describen se deben ejecutar pruebas de seguridad y control de acceso (Ver epígrafe 2.8.2.1)

3.3.1 Caso de prueba Nueva HSF

Descripción

El caso de prueba se inicia cuando un probador desea crear una HSF Básica, selecciona en el menú la opción Nueva HSF y el sistema le muestra los parámetros que debe llenar, entre ellos la dirección de la vivienda a la cual se le va a asignar la nueva HSF Básica, que se obtiene mediante una búsqueda en el RAS (Registro de Área de Salud). El probador introduce los datos solicitados omitiendo la dirección.

Condiciones de ejecución

Se debe haber hecho la búsqueda de vivienda en el RAS de forma exitosa.

Entradas

Fecha: 2007-05-30
Tipo de HSF: Básica
Dirección: Vacío

Resultado esperado

Muestra un mensaje: EXISTE ALGÚN CAMPO VACÍO

3.3.1.1 Caso de prueba Nueva HSF

Descripción

El caso de prueba se inicia cuando un probador desea crear una HSF Básica, selecciona en el menú la opción Nueva HSF y el sistema le muestra los parámetros que debe llenar. El probador introduce los datos solicitados y el sistema crea la nueva HSF Básica asociándole un identificador y muestra un listado de las HSF Básicas actualizado.

Condiciones de ejecución

Se debe haber hecho la búsqueda de vivienda en el RAS de forma exitosa.

Entradas

Fecha: 2007-05-30
Tipo de HSF: Básica
Dirección: s/n 7, calle 10, entre 3 y 9, localidad Castillo de Jagua

Resultado esperado

Muestra los datos de la HSF creada:
Tipo: Básica

Dirección: s/n 7, calle 10, entre 3 y 9, localidad Castillo de Jagua
No HSF (se genera a partir de la dirección)

3.3.2 Caso de prueba Buscar Vivienda

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Nueva HSF, dentro de esta la búsqueda de una vivienda, la cual se realiza en el RAS, y el sistema le muestra los parámetros de búsqueda que debe llenar. El probador introduce los datos solicitados.

Condiciones de ejecución

Debe existir la dirección en el RAS y sin HSF Básica asociada.

Entradas

Localidad: Castillo de Jagua
Consejo Popular: No 1 Jagua 121
Circunscripción: 145
Zona CDR: 3
CDR: 1 Pedro A Pérez

Resultado esperado

Lista la vivienda:
Calle 10, No 14, Entre 2 y 3, Localidad Castillo de Jagua, CDR 1 Pedro A Pérez

3.3.3 Caso de prueba Buscar HSF

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Buscar HSF, el sistema muestra los parámetros de búsqueda para la HSF Básica y el probador selecciona los parámetros por los que desea realizar la búsqueda.

Condiciones de ejecución

Debe existir al menos una HSF Básica.

Entradas

Tipo HSF: Básica
Localidad: El Desvío
Población: Población 9

Resultado esperado

No HSF: 101410181
Tipo: Básica
Dirección: No 17, Calle Sur, El Desvío.
No HSF: 1014109510
Tipo: Básica
Dirección: No 33, Calle Sur, El Desvío.

3.3.4 Caso de prueba Buscar Ciudadano

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Insertar paciente para agregar en una HSF Básica, un paciente que no se encuentra en el Registro de Ciudadano, el sistema muestra la

búsqueda en este Registro y el probador selecciona los parámetros por los que desea realizar la búsqueda.

Condiciones de ejecución

Debe haberse realizado la búsqueda de la HSF Básica No 1214516 exitosamente y no debe existir el ciudadano en el Registro Ciudadano.

Entradas

CI: 84101421713
Nombre: Celia
Apellido: Pérez
Fecha de nacimiento: 1984-10-14

Resultado esperado

Al no encontrar el paciente, el sistema debe brindar la opción de insertar el paciente en el Registro de Ciudadano luego de la búsqueda e insertarlo.

3.3.5 Caso de prueba Insertar en Ciudadano

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Insertar paciente para agregar un paciente que no se encuentra en el Registro de Ciudadano en una HSF Básica, el sistema muestra la búsqueda en este Registro y luego la opción de insertarlo.

Condiciones de ejecución

Debe haberse realizado la búsqueda de la HSF Básica No 1214516 exitosamente y la búsqueda en Ciudadano debe haber sido exitosa al no encontrar al paciente

Entradas

CI: 63021234567
Nombre: Magdalena
Apellido: Basulto
Fecha de nacimiento: 1963-02-12

Resultado esperado

Inserta al paciente y muestra los datos entrados.

3.3.6 Caso de prueba Insertar Paciente

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Insertar paciente para agregar un paciente que tiene HSF Básica asociada, en una nueva HSF Básica, el sistema muestra la búsqueda en el Registro de Ciudadano y se encuentra al paciente en el mismo.

Condiciones de ejecución

Debe haberse realizado la búsqueda de la HSF Básica No 1014109011 exitosamente y la búsqueda en Ciudadano debe haber sido exitosa al encontrar al paciente en el mismo.

Entradas

Nombre: Roberto
Apellido: Del Castillo
Sexo: Masculino
Nivel Educacional: Universitario
Labor que realiza: Director General de Empresa
Profesión: Estomatólogo
No HSF Individual: 85120578452

Resultado esperado

Muestra un mensaje: “ESTE PACIENTE YA PERTENECE A UNA HSF BÁSICA”.

3.3.7 Caso de prueba Insertar Ingreso

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Insertar ingreso para agregar un ingreso a un paciente que tiene HSF Básica asociada, el sistema muestra los criterios por los cuales se hará el ingreso y el probador introduce la información.

Condiciones de ejecución

Debe haberse realizado la búsqueda de la HSF Básica No 1014109011 exitosamente, seleccionado dentro de esta HSF Básica el paciente Roberto del Castillo González y encontrado en RPSAP (Registro de Problemas de Salud) la enfermedad por la cual se realiza el ingreso.

Entradas

Fecha Inicio: 2007-06-05
Causa: Hepatitis aguda Tipo A, con coma hepático.

Resultado esperado

Se muestra los datos del paciente en el header o encabezado de la página: HC Individual, Nombre y Apellidos, Sexo, Nivel Educacional, Profesión, Labor que realiza, Grupo Dispensarial y Edad.

Además se debe mostrar el nuevo ingreso con los siguientes datos:

Fecha de inicio: 2007-06-05
Fecha de fin: Pendiente
Causa: Hepatitis aguda Tipo A, con coma hepático.

3.3.8 Caso de prueba Insertar Seguimiento Diario

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Insertar Seguimiento Diario para agregar un seguimiento a un ingreso que tiene un determinado paciente, el sistema muestra los campos de entrada y el probador introduce una fecha anterior a la fecha de inicio del ingreso.

Condiciones de ejecución

Debe haberse realizado la búsqueda de la HSF Básica No 1014109011 exitosamente, seleccionado dentro de esta HSF Básica el paciente Roberto del Castillo González y el ingreso insertado en el caso de prueba anterior, que es al que la razón por la cual se decide hacer el seguimiento diario

Entradas

Fecha Inicio: 2007-06-04
Observación: Evolución exitosa

Resultado esperado

El sistema muestra un mensaje: “FECHA NO VÁLIDA”.

3.3.9 Caso de prueba Agregar Funcionamiento Familiar

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Agregar Funcionamiento Familiar para evaluar el funcionamiento familiar de una HSF Básica determinada, el sistema muestra los criterios por los cuales se evaluará el Funcionamiento Familiar y el probador introduce la información.

Condiciones de ejecución

Debe haberse realizado la búsqueda de la HSF Básica No 1014109011 exitosamente.

Entradas

Fecha de evaluación: 2007-06-30
Funcionalidad: Funcional

Resultado esperado

Se muestra en el header o encabezado de la página la información perteneciente a la HSF Básica que se le está evaluando el Funcionamiento Familiar: No de la HSF Básica, Dirección y Área de Salud que la atiende.

Además muestra:

Fecha de evaluación: 2007-06-30
Funcionalidad: Funcional

3.3.10 Caso de prueba Buscar Integrante

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Buscar Integrante para buscar una persona que ya pertenece a una HSF Básica, el sistema muestra los criterios por los cuales se desea hacer la búsqueda y el probador selecciona la opción Buscar Todos.

Condiciones de ejecución

Debe existir la HSF Básica asociada a esa persona y por tanto la persona dentro de esta HSF Básica.

Entradas

Sexo: Masculino

Resultado esperado

Se muestran todos los datos del paciente encontrado:

CI: 85120578452
No HSF Básica: 1014109011
Nombre: Roberto
1er Apellido: Del castillo
2do Apellido: González
Sexo: Masculino
Edad: 21
Grupo Dispensarial: Grupo I

3.3.11 Caso de prueba Agregar Condición de Vida Familiar

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Agregar Condición de Vida Familiar para evaluar esta opción a una HSF Básica, el sistema muestra los criterios por los cuales se puede evaluar y el probador introduce la información.

Condiciones de ejecución

Debe haberse realizado la búsqueda de la HSF Básica No 1014109011 exitosamente.

Entradas

Fecha de Evaluación: 2007-06-05
Condición Estructural Vivienda: Bien
Índice de Hacinamiento: Alto
Equipamiento Doméstico Básico: Bueno
Satisfacción con Ingresos: Medianamente Satisfechos

Resultado esperado

Se muestra en el header o encabezado de la página la información perteneciente a la HSF Básica que se le está evaluando el Funcionamiento Familiar: No de la HSF Básica, Dirección, Área de Salud, Grupo Básico de Trabajo (GBT) y Equipo Básico de Salud (EBS) que la atiende.

Además muestra cómo quedo la evaluación hecha:

Fecha de Evaluación: 2007-06-05
Condición Estructural Vivienda: Bien
Índice de Hacinamiento: Alto
Equipamiento Doméstico Básico: Bueno
Satisfacción con Ingresos: Medianamente Satisfechos

3.3.12 Caso de prueba Agregar Evaluación de Salud Familiar

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Agregar Evaluación de Salud Familiar para evaluar esta opción a una HSF Básica, el sistema muestra los criterios por los cuales se puede evaluar y el probador introduce la información.

Condiciones de ejecución

Debe haberse realizado la búsqueda de la HSF Básica No 1014109011 exitosamente.

Entradas

Fecha de Evaluación: 2007-06-05
Evaluación: Con Problemas de Salud de Integrantes de la familia

Resultado esperado

Se muestra en el header o encabezado de la página la información perteneciente a la HSF Básica que se le está evaluando el Funcionamiento Familiar: No de la HSF Básica, Dirección, Área de Salud, Grupo Básico de Trabajo (GBT) y Equipo Básico de Salud (EBS) que la atiende.

Además muestra cómo quedo la evaluación hecha:

Fecha de Evaluación: 2007-06-05
Evaluación: Con Problemas de Salud de Integrantes de la familia

3.3.13 Caso de prueba Agregar Intervención Familiar

Descripción

El caso de prueba se inicia cuando un probador selecciona la opción Agregar Intervención Familiar para evaluar esta opción a una HSF Básica, el sistema muestra los criterios por los cuales se puede evaluar y el probador introduce la información.

Condiciones de ejecución

Debe haberse realizado la búsqueda de la HSF Básica No 1014109011 exitosamente.

Entradas

Fecha de Intervención: 2007-06-05
Tipo de Intervención: Intervención Terapéutica

Resultado esperado

Se muestra en el header o encabezado de la página la información perteneciente a la HSF Básica que se le está evaluando el Funcionamiento Familiar: No de la HSF Básica, Dirección, Área de Salud, Grupo Básico de Trabajo (GBT) y Equipo Básico de Salud (EBS) que la atiende.

Además muestra cómo quedo la evaluación hecha:

Fecha de Intervención: 2007-06-05
Tipo de Intervención: Intervención Terapéutica

3.4 Evaluación de los resultados

3.4.1. Caja blanca

Los resultados de la ejecución de los casos de pruebas de caja blanca para la opción Insertar Problemas de Salud fueron satisfactorios al lograr insertar los problemas de salud deseados y mostrando los grupos dispensariales actualizados luego de las inserciones y cumpliendo así parte de los resultados esperados. Pero hay error en el mensaje que se muestra cuando se desea insertar un problema de salud que ya existe: “Ya existe ese problema de salud”, el cual debe estar en letra mayúscula según lo establecido en los estándares de codificación definidos por el Proyecto SISalud. Existe una doble validación de los parámetros de entrada en el código, que aunque no repercute en la obtención de un resultado exitoso, influye en la obtención de un código óptimo que en un futuro puede ser reutilizado por otro desarrollador o un miembro del equipo de prueba.

Se debe dar mantenimiento a la validación del negocio ValidarEnteroPositivo, excluir el cero de los enteros positivos pues cuando se introduce un parámetro que no es entero desde la capa de presentación para que en el negocio no se tome como una cadena (string) , hay que forzarlo a entero, cuando esto sucede, la variable de sesión toma valor cero, la función de validar lo considera entero positivo y no muestra el mensaje de error en capa de presentación siendo el parámetro incorrecto y por último la validación de caracteres extraños debía mostrar el mensaje: “ALGÚN PARÁMETRO DE ENTRADA CONTIENE CARACTERES EXTRAÑOS” y mostró el mismo mensaje que se muestra cuando se omite algún parámetro de entrada.

3.4.2 Caja negra

3.4.2.1 Caso de prueba Nueva HSF

Los resultados de la ejecución de este caso de prueba fueron satisfactorios, se valida correctamente los campos que deben ser llenados obligatoriamente mostrando los mensajes correspondientes, aunque se debe señalar que el hint del botón aceptar de esta opción es agregar causa de ingreso y debería ser agregar nueva HSF, en correspondencia con la funcionalidad.

3.4.2.2 Caso de prueba Nueva HSF

Los resultados de la ejecución de este caso de prueba fueron exitosos al crear la nueva HSF Básica a partir de una dirección previamente buscada pero demuestra un error al generar el No de HSF Básica que debe ser generado a partir de la dirección de la vivienda.

3.4.2.3 Caso de prueba Buscar Vivienda

Los resultados de la ejecución de este caso de prueba demuestran errores en la funcionalidad de la opción Buscar Vivienda al no listar ninguna vivienda a partir de los criterios de búsqueda especificados, siendo diferente los resultados esperados. Además cuando se busca una vivienda y ya todas tienen asignadas HSF Básica, el sistema debería mostrar un mensaje avisando al usuario que no existen viviendas sin asignarle HSF Básica. En los parámetros de búsqueda se debería validar la relación que existe entre las diferentes partes que conforman una dirección de una vivienda, como por ejemplo: luego de seleccionar una manzana debe existir un límite de entre calles para seleccionar.

3.4.2.4 Caso de prueba Buscar HSF

Los resultados de la ejecución de este caso de prueba son satisfactorios tal y como lo plantearon los resultados que se previeron aunque se debe señalar que las validaciones de los parámetros de entrada no están hechas.

3.4.2.5 Caso de prueba Buscar Ciudadano

Los resultados de la ejecución de este caso de prueba demuestran errores en la funcionalidad de esta opción teniendo en cuenta que los nombres y apellidos del ciudadano que se deseaba buscar en el Registro de Ciudadano ya se encontraban en la Base de Datos de Ciudadano pero se le especificó un

número de carne de identidad (CI) diferente, esto demuestra que no buscó por todos los parámetros de entrada, siendo el CI el más importante pues es una identificación única. Mostró un mensaje de la existencia del ciudadano, y a pesar de esto lo insertó en el Registro de Ciudadano, una completa contradicción. Además no se distingue la funcionalidad entre los botones Agregar y Aceptar, y este último no tiene función implementada, mostrando una pagina de error cuando se selecciona.

3.4.2.6 Caso de prueba Insertar en Ciudadano

Los resultados de la ejecución de este caso de prueba son satisfactorios al lograr insertar el ciudadano que se deseaba aunque se debe especificar que es necesario establecer un límite de caracteres para el nombre y apellido, tal y como se hace para los números de HSF Básica e individual y por otro lado la fecha de nacimiento que esta dentro de los parámetros de entrada debería generarse a partir del CI introducido, teniendo en cuenta que de todas formas valida la fecha de nacimiento según el CI para saber si está correcta.

3.4.2.7 Caso de prueba Insertar paciente

El resultado de la ejecución de este caso de prueba muestra errores en la funcionalidad de esta opción teniendo en cuenta que no se verifica si el paciente que se desea insertar a una HSF Básica tiene asociada otra HSF Básica y se debería mostrar un mensaje informándole esta situación al probador. En este caso se insertó un paciente a una HSF Básica, teniendo otra HSF Básica asignada sin problema alguno. Además se debe tener presente cuando se inserta un paciente a una HSF Básica que si es por segunda vez, en el menú de causa ya no deben salir las opciones: censo y recién nacido, sólo debe salir la opción traslado. De lo contrario si se inserta por primera vez sí se deben mostrar las tres opciones.

3.4.2.8 Opción Insertar Ingreso

Los resultados de la ejecución de este caso de prueba son satisfactorios teniendo en cuenta que se agregó el ingreso con los datos deseados pero además no mostró los datos del paciente al que se le está visualizando las causas de ingreso, dejando información vacía que es necesaria para el usuario. Y los botones Exportar a, Agregar, y Eliminar no tienen funcionalidad.

3.4.2.9 Opción Insertar Seguimiento Diario

Los resultados de la ejecución de este caso de prueba son satisfactorios teniendo en cuenta los resultados esperados pues se mostró un mensaje de error partiendo de un dato no válido introducido al seleccionar la fecha del seguimiento diario anterior a la fecha del ingreso, esta situación es imposible ya que el ingreso es lo que motiva la necesidad de realizar un seguimiento.

Y como sugerencias se tienen:

El text area donde se introduce las observaciones que hace el miembro del equipo básico durante el seguimiento diario debe ser mas amplio para permitir que se visualice la información que se esta escribiendo. Además debe limitarse la cantidad de caracteres de las observaciones pues un tamaño exagerado deforma la configuración de la página donde se muestra el resultado. No muestra el número de la HSF Básica en el encabezado de la página y muestra dos mensajes que de forma diferente trasmiten el mismo mensaje cuando se omiten las observaciones.

3.4.2.10 Opción Insertar Funcionamiento Familiar

Los resultados de la ejecución de este caso de prueba son satisfactorios tal y como lo plantearon los resultados que se previeron, pues se agregó la evaluación del Funcionamiento Familiar exitosamente.

3.4.2.11 Opción Buscar Integrante

Los resultados de la ejecución de este caso de prueba son satisfactorios en correspondencia con los resultados esperados especificados en el mismo, pues de la búsqueda resultaron los pacientes que tenían asociado HSF Básica.

3.4.2.12 Opción Agregar Condición de Vida Familiar

Los resultados de la ejecución de este caso de prueba son satisfactorios tal y como lo plantearon los resultados que se previeron, pues se agregó la evaluación de las Condiciones de Vida Familiar y se mostró toda la información pertinente luego de la inserción.

3.4.2.13 Opción Agregar Evaluación de Salud Familiar

Los resultados de la ejecución de este caso de prueba son satisfactorios tal y como lo plantearon los resultados que se previeron, pues se agregó la evaluación de la Salud Familiar y se mostró toda la información pertinente luego de la inserción.

3.4.2.14 Opción Agregar Intervención Familiar

Los resultados de la ejecución de este caso de prueba son satisfactorios tal y como lo plantearon los resultados que se previeron, pues se agregó la Intervención Familiar y se mostró toda la información pertinente luego de la inserción.

Además de estos casos de prueba hay errores y sugerencias de otras funcionalidades que se especifican seguidamente.

3.4.2.15 Opción Problemas de Salud

No muestra el número de la HSF Básica, que forma parte de la información que junto a los problemas de salud debe mostrarse.

3.4.2.16 Opción Eliminar paciente

Debe tener un mecanismo que exija seleccionar un paciente de una HSF Básica para poder eliminarlo, lo más adecuado sería un mensaje: DEBE SELECCIONAR UN PACIENTE.

Uno de los atributos de calidad más relevantes para aplicaciones WEB definidos por Roger S. Pressman es la usabilidad en la cual se mide la capacidad de comprensión del sitio global.

Durante el proceso de prueba se detectaron algunos aspectos que pueden mejorarse con el objetivo de lograr una aplicación más usable.

En la opción Nueva HSF luego de crear una HSF Básica muestra el listado de todas las HSF básicas que se han creado, por lo cual se considera que es más factible que muestre sólo la HSF Básica que se creó para seguidamente agregar pacientes o demás datos propios de la gestión de una HSF Básica, o mostrar la HSF Básica creada con la posibilidad de seguir creando otras HSF Básicas.

En la opción Buscar en el Registro de Ciudadano cuando se introduce algún parámetro incorrecto y muestra un mensaje de error, aunque se salga de la opción y se vuelva entrar no se actualizan los datos. Este aspecto afecta de manera general a toda la aplicación, resultando necesario en la mayoría de los casos salir del módulo RPOB y volver a entrar para ejecutar la opción deseada.

3.5 Prueba de Interfaz de usuario

De manera general la interfaz permite el acceso y navegación a todas las funcionalidades permitidas de acuerdo al tipo de usuario mediante los menús, así como la función de Ayuda Online que brinda la aplicación. Pero además se detectaron palabras incompletas en los mensajes y acentos omitidos que aunque no repercuten a un alto grado en la buena apariencia de la interfaz sí influyen en que cada interfaz sea amigable al usuario. Y se omitieron validaciones de campos que son necesarias para un buen funcionamiento de cada opción.

3.6 Prueba de integridad de los datos y a la Base de Datos

Mediante la incorporación de datos de prueba con el script generado se comprobó que las inserciones, modificaciones y eliminaciones se ejecutaron exitosamente y que la información de la Base de Datos se actualiza y guarda adecuadamente según los cambios que presupone cada ejecución de las funcionalidades.

3.7 Pruebas de Seguridad y Control de acceso

Se verificó el acceso al módulo RPOB y según el tipo de usuario, nacional, provincial, o municipal, los permisos que le son concedidos. Además los mecanismos de seguridad al intentar acceder a un método sin el adecuado permiso, lo cual se garantiza con la función plaser_handler que se instancia en cada método de RPOB.

3.8 Resumen de Evaluación

Luego de evaluar cada funcionalidad por separado se considera que se han identificado y registrado todos los errores de la opción HSF Básica así como sugerencias en determinados aspectos con el objetivo de lograr mejoras en un producto que finalmente va a ser evaluado por su destino, que es el usuario.

Errores comunes para varias funcionalidades:

- Los botones **Exportar a** no tienen funcionalidad en el página donde se encuentran.
- Cuando se ejecuta una opción a parte del resultado de las inserciones o modificaciones se debe mostrar en el encabezado o header de la página información del paciente o de la HSF Básica que se está modificando, lo cual en muchos casos se viola, mostrando parte de esa información vacía.
- Los resultados de las páginas no se actualizan, siendo necesario en muchas ocasiones salir de la aplicación y volver a entrar para se refresque la interfaz, esto es muy frecuente cuando alguna opción muestra mensajes de error.

Errores específicos:

- **Nueva HSF**: El hint del botón aceptar de esta opción es agregar causa de ingreso y debe ser agregar nueva HSF, en correspondencia con la funcionalidad.
- **Buscar Vivienda**: En los parámetros de búsqueda no se valida la relación que existe entre las diferentes partes que conforman una dirección de una vivienda, como por ejemplo: luego de seleccionar una manzana debe existir un límite de entre calles para seleccionar.
- **Buscar HSF**: No se validan los parámetros de entrada.

- **Buscar Ciudadano**: No se distingue la funcionalidad entre los botones Agregar y Aceptar, este último no tiene funcionalidad por lo que muestra una página de error cuando se selecciona.
- **Insertar Paciente**: No se verifica si el paciente que se desea insertar a una HSF Básica tiene asociada otra HSF Básica y se debe mostrar un mensaje informándole esta situación al probador.
- **Insertar Seguimiento Diario**: Muestra dos mensajes que de forma diferente trasmiten el mismo mensaje cuando se omiten las observaciones.
- **Eliminar Paciente**: Debe tener un mecanismo que exija seleccionar un paciente de una HSF Básica para poder eliminarlo, lo más adecuado sería un mensaje: DEBE SELECCIONAR UN PACIENTE.

Errores en las funcionalidades:

- **Buscar Vivienda**
- **Buscar en Ciudadano**
- **Insertar Paciente**

La explicación de estos errores se describe en los epígrafes **3.4.2.3**, **3.4.2.5** y **3.4.2.7** respectivamente.

Sugerencias comunes:

Debe limitarse la cantidad de caracteres que se introduce en las observaciones y las descripciones pues debe ser una síntesis y de ser demasiado extensa deforma la configuración de la página donde se muestran los resultados.

3.9 Sugerencias específicas

Buscar Vivienda: Cuando se busca una vivienda y ya todas tienen asignadas HSF Básica, el sistema debería mostrar un mensaje avisando al usuario.

Insertar en Ciudadano: La fecha de nacimiento que esta dentro de los parámetros de entrada debería generarse a partir del CI introducido, teniendo en cuenta que de todas formas valida la fecha de nacimiento según el CI para saber si está correcta.

Insertar Paciente: Se debe tener presente cuando se inserta un paciente a una HSF Básica que si es por segunda vez, en el menú de causa ya no deben salir las opciones: censo y recién nacido, sólo debe salir la opción traslado. De lo contrario si se inserta por primera vez si se deben mostrar las tres opciones.

Insertar Seguimiento Diario: El text area donde se introduce las observaciones que hace el usuario durante el seguimiento diario debe ser mas amplio para permitir que se visualice la información que se esta escribiendo.

Insertar problemas de Salud: Se debe eliminar una de las dos validaciones de los parámetros de entrada.

Con la realización de este capítulo se puede arribar a algunos consejos importantes a la hora de diseñar buenos casos de prueba y ejecutarlos:

El programador debe evitar probar sus propios programas ya que desea (consciente o inconscientemente) demostrar que funcionan sin problemas. Además que es normal que las situaciones que olvidó al crear el programa queden de nuevo olvidadas al crear los casos de prueba.

Al generar casos de prueba se debe incluir tantos datos de entrada válidos y esperados como no válidos e inesperados.[10]

[10] Collado,M.,pruebas de Software.2003 [Disponible en http://lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt]

CONCLUSIONES

Con la realización de este trabajo de diploma se arribaron a las siguientes conclusiones:

- Se ejecutó el flujo de trabajo de prueba aplicando las buenas prácticas del Proceso de Software Personal (PSP), Proceso de Software en Equipo (TSP) y de la disciplina de pruebas del proceso de Ingeniería de Software.
- Se definió el plan de prueba especificando fundamentalmente el alcance, la estrategia y los recursos.
- Se ejecutaron y evaluaron 19 casos de prueba diseñados y se obtuvo la lista de defectos con un resumen de errores y sugerencias.
- Se generó un componente de prueba y un script de datos de prueba.

De forma general, el flujo de de trabajo de prueba se ejecutó de forma exitosa partiendo de que el éxito de las pruebas se encuentra en la detección de errores. Teniendo en cuenta que el proceso de prueba se hizo paralelo al desarrollo del módulo Registro de Población todos los errores encontrados se informaron y fueron arreglados, colaborando de esta manera con el proceso de Gestión de Cambios, se cumplió con los objetivos propuestos ya que en la actualidad la opción Historia de Salud Familiar (HSF) del Registro de Población tiene una mayor calidad y un alto nivel técnico.

RECOMENDACIONES

Al concluir con los objetivos propuestos en la investigación se recomienda al proyecto SISalud:

- Implementar un componente de prueba configurable que automatice el proceso de ejecución de varios casos de prueba.
- Utilizar el script de datos de prueba para enriquecerlo con datos de prueba de otros módulos.
- Aplicar este flujo de trabajo en otros módulos del proyecto para evaluarlo.
- Extender el flujo de trabajo de prueba al resto de las opciones del módulo RPOB.
- Desarrollar en próximos módulos del proyecto, un proceso de prueba que abarque todas las fases del proceso de desarrollo de un software.

BIBLIOGRAFÍA

1. B., J. Goodenough. Team Software Process Reliability Results. 2000 [Disponible en: http://www.softwaretechnews.com/stn3-4/teamspi.html].
2. Beizer, B. Software Testing Techniques. 2a ed. 1990.
3. Booch, G., Rumbaugh, J., Jacobson, I. El Proceso Unificado de Desarrollo de Software. La Habana, Cuba 2004.
4. Colectivo de Profesores. Prueba. 2004 [Disponible en: http://ucimedia.uci.cu/teleclases/2005-2006/2do-sem/mc/ingenieria_de_software_2_04-05/conf3].
5. Colectivo de Profesores. Fase de Elaboración. Flujo de trabajo de prueba.2005 [Disponible en: http://ucimedia.uci.cu/teleclases/2005-2006/2do-sem/3er/ingenieria_de_software_2/conf3].
6. Colectivo de Profesores. Fase de Construcción. Flujo de trabajo de prueba.2005 [Disponible en: http://ucimedia.uci.cu/teleclases/2005-2006/2do-sem/3er/ingenieria_de_software_2/conf4].
7. Colectivo de Profesores. Culminación de la Fase de Elaboración. Flujo de trabajo de Prueba. 2005
8. Collado,M.,pruebas de Software. 2003 [Disponible en: http://lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt].
9. Gonzalez,C.,Un Plan de Pruebas Exitoso.2002 [Disponible en http://www.americaxxi.cl/modules.php?name=News&file=article&sid=20].
10. Humphrey, W. S. Introducción al Proceso de Software en Equipo. 2001.
11. Humphrey, W. S. Introducción al Proceso Software Personal. Madrid 2001.
12. Jacobson, I., Booch, G. y Rumbaugh, J. El Proceso Unificado de Desarrollo de Software. 2000.
13. Mañas, J. Prueba de Programas. 1994 [Disponible en: http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#sA].
14. Marin M. Consideraciones sobre el proyecto de informatización de la Atención Primaria de Salud. Editorial. Revista Habanera de Ciencias Médicas. Vol 3. No. 10. año 2004. [Disponible en: http://www.ucmh.sld.cu/rhab/editorial_rev10.htm].
15. Marin M. Capacitación en el área de la Informática en Salud. Revista Cubana de Informática Médica (RCIM).

16. Montes de Oca, C. V. Team Software Process (TSP): Integración de Equipos de Desarrollo de Alto Rendimiento. 2007 [Disponible en: http://www.emagister.com/el-team-software-process-tsp-cursos-1052582.htm]
17. Padrón, L. A., Informática para la atención primaria de salud. VI Congreso Internacional de Informática en Salud. 2007
18. Pressman, R. S. Ingeniería del Software. Un enfoque práctico.5 ed. 2001, Madrid.
19. Pressman, R. S. Ingeniería del Softawre. Un enfoque práctico. La Habana, Cuba 2005.
20. Roca, M.V., Pruebas Funcionales. Aspectos relevantes.2006 [Disponible en: http://www.greensqa.com/archivos/Art02-PlaneacionPruebasFuncionales.pdf].

GLOSARIO DE TÉRMINOS

Dispensarización: Clasificación de los pacientes que están registrados en la HSF, de acuerdo a la enfermedad, en Grupos Dispensariales.

Grupos Dispensariales: Grupos establecidos por el MINSAP (Ministerio de Salud Pública), Aparentemente sano, Riesgo, Enfermedad y Discapacidad.

HSF: Historia de Salud Familiar

PSP: Proceso de Software Personal

RAS: Registro de Área de Salud

RE: Registro de Estudiantes

RUS: Registro de Unidades de Salud

RL: Registro de Localidades

RPOB: Registro de Población

RPS: Registro del Personal de la Salud

RSM: Registro de Servicios Médicos

RPSAP: Registro de Problemas de Salud para la Atención Primaria

RCIE: Registro de Clasificación Internacional de Enfermedades

RC: Registro de Ciudadano

RIS: Registro Informatizado de la Salud

SAAA: Seguridad basada en el modelo de Autenticación, Autorización y Auditoría

SISalud: Sistema de Información para la Salud

TSP: Proceso de Software en Equipo